VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA
EKONOMICKÁ FAKULTA

KATEDRA APLIKOVANÉ INFORMATIKY

Tvorba internetového obchodu s využitím frameworku Nette

E-shop System based on Nette Framework

Student: Jaroslav Klimčík

Vedoucí bakalářské práce: Ing. Pochyla Martin, Ph.D.

Ostrava 2013

VŠB - Technická univerzita Ostrava
Ekonomická fakulta
Katedra aplikované informatiky

Zadání bakalářské práce

Student: **Jaroslav Klimčík**

Studijní program: B6209 Systémové inženýrství a informatika

Studijní obor: 6209R001 Aplikovaná informatika

Téma: Tvorba internetového obchodu s využitím frameworku Nette
E-shop System Based on Nette Framework

Zásady pro vypracování:

1. Úvod
2. Teoretická východiska tvorby internetové aplikace
3. Analýza a návrh řešení
4. Realizace a implementace internetového obchodu
5. Závěr

Seznam použité literatury
Seznam zkratek
Prohlášení o využití výsledků bakalářské práce
Seznam příloh
Přílohy

Seznam doporučené odborné literatury:

LECKY-THOMPSON, Ed a Steven D. NOWICKI. *PHP 6: programujeme profesionálně.* Překlad Ondřej GIBL. Brno: Computer Press, 2010. ISBN 978-80-251-3127-5.
VAN DUYNE, D. K., J. A. LANDAY a J. I. HONG. *Návrh a tvorba webů: vytváříme zákaznicky orientovaný web.* Brno: CP Books, 2005. ISBN 80-251-0508-3.
SCHLOSSNAGLE, George. *Pokročilé programování v PHP 5.* Brno: Zoner Press, 2004. ISBN 80-868-1514-5.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Pochyla, Ph.D.**

Datum zadání: 23.11.2012
Datum odevzdání: 10.05.2013

Ing. Petr Rozehnal, Ph.D.
vedoucí katedry

prof. Dr. Ing. Dana Dluhošová
děkanka fakulty

Prohlašuji, že jsem celou práci, včetně příloh, vypracoval samostatně.

.....

Jaroslav Klimčík

V Ostravě dne 10.5.2013

Rád bych poděkoval vedoucímu práce panu Ing. Martinu Pochylovi, Ph.D. za jeho cenné rady, vstřícnost a čas, který mi byl z jeho strany poskytnut. Chtěl bych také poděkovat i těm, kteří mi jakkoliv pomohli při zpracování této bakalářské práce.

Obsah

1. Úvod

Termín internet dnes již není žádnou velkou neznámou jak tomu bývalo v dřívějších letech. Jeho obliba rapidně roste a lidé ho stále více využívají při své práci, relaxaci nebo běžných činnostech, jako je například online nakupování. S větší oblibou používání internetu roste i větší počet potenciálních zákazníků. To je také důvodem, proč firmám přestává stačit pouhá prezentace na internetu a zvyšuje se poptávka po systémech pro internetové obchody. Kromě stále se zvyšujícího počtu zákazníku, přesouvají společnosti své podnikání na internet za účelem snižování svých nákladů, zrychlení transakčních procesů, rychlejší odezvy zákazníkovi a zefektivnění svých služeb. Elektronické obchody, neboli e-shopy, tak mohou vykazovat trvalé zisky vycházející z možnosti doporučovat zákazníkům nové produkty v závislosti na jejich předchozích objednávkách a také díky neustále dostupnému aktuálnímu katalogu produktů pro zákazníky.

Protože v současnosti existuje již nespočet e-shopů, je těžké být mezi touto konkurencí napřed a je velmi důležité, aby především menší podnikatelé věděli, jak přilákat zákazníky. Nakupování v kamenných obchodech pomalu klesá a naopak se zvyšuje nakupování přes internet. Vyvíjejí se moderní aplikace, které se snaží zákazníkovi vyjít maximálně vstříc a ulehčit mu nakupování od volby výrobku až po jednoduchou dopravu až domů. Aby se elektronický obchod odlišoval od konkurence, je potřeba aby zákazníci ocenili pohodlí online objednávky a obchod působil dojmem, že nabízí osobní výhody, díky kterým si zákazníky udrží. Pro originalitu a odlišitelnost od jiných obchodů, je důležité vytvořit jedinečný design, který bude přehledný, nabídne mnoho funkcí, na potenciální zákazníky zapůsobí věrohodně a díky tomu je upoutá a přesvědčí k nákupu některého z produktů.

V současnosti si můžeme nechat vytvořit unikátní nebo univerzální e-shop od specializované firmy, nicméně cena za opravdu kvalitní elektronický obchod, se pohybuje ve vysokých částkách. Na druhou stranu existují tzv. open-source e-shopy, jako například Magento, PrestaShop nebo CubeCart, které nabízejí své služby zadarmo. Obě varianty nabízí zájemci o vlastní e-shop relativně jednoduché a rychlé řešení. Problém však nastává, pokud chce vlastník obchodu něco přidat, změnit či odebrat. U placené formy formy elektronického obchodu to znamená výdej dalších financí, které mohou dosahovat vyšších částek. U open-source řešení to může být rovněž problém, především pak co se týče vzhledu. Ten je systému

„šitý“ na míru a jeho změna může vyžadovat specialistu tohoto systému nebo nutnost použít úplně jinou šablonu, která však již nemusí obsahovat prvky, které vlastník požaduje.

Třetí variantou je pak vytvoření vlastního obchodu. Toto řešení je velmi flexibilní co se týče požadavků klienta. Umožňuje navrhovat unikátní vzhled, kde není problém změna určitého prvku či použití úplné jiného vzhledu či šablony. V případě, že by zákazník chtěl použít některou funkci z jiného e-shopu, není takový problém funkcionalitu obchodu rozšířit podle požadavků.

Cílem mé bakalářské práce je tvorba internetového obchodu s využitím frameworku Nette, který bude obsahovat přívětivé uživatelské rozhraní a jednoduchou administraci, kde se bude odehrávat správa celé aplikace. Zároveň bude splňovat všechny požadavky klienta. Framework umožní s minimálním úsilím rozšířit systém pro potřeby jakéhokoli projektu elektronického obchodu. Použití Nette Frameworku ulehčí mnoho částí vývoje, které bychom bez něj museli obtížně řešit.

2. Teoretická východiska tvorby internetové aplikace

V této kapitole jsou uvedeny všechny technologie, které budou použity při realizaci a implementaci internetového obchodu. Všechny tyto technologie jsou open-source a v prostředí Internetu jsou hojně využívány. Ve druhé části kapitoly jsou popsány použité technologie pro modelování, které jsou dále použity v analýze a návrhu řešení. Poslední část je věnovaná bezpečnosti aplikace.

2.1 HTML

HTML, neboli HyperText Markup Language, je značkovací jazyk, který slouží k tvorbě internetových stránek a jejich prezentaci v internetovém prohlížeči. Cílem internetové prohlížeče tak je číst HTML dokumenty a skládat je do vizuálních (případně zvukových) webových stránek. Prohlížeče nezobrazují HTML tagy, ale využívá je k interpretování obsahu stránky.

HTML tak tvoří základní kameny pro všechny webové stránky. Umožňuje vkládat obrázky či různé objekty, které pak mohou být využity například pro interaktivní formuláře. HTML díky strukturální sémantice vytváří strukturální dokumenty, které pak mohou obsahovat nadpisy, odstavce, seznamy, odkazy a jiné prvky.

2.1.1 Historie HTML

Historie HTML započala roku 1989, kdy si Tim Berners-Lee, pracovník ve švýcarském CERNu, uvědomil, že musí existovat lepší a rychlejší způsob, jak získávat soubory informací, aniž by se musel stále přihlašovat a odhlašovat na různých počítačích. Přišel tedy s konceptem navrhující internetový hypertextový systém. O rok později specifikoval jazyk HTML, jenž tvoří základ Word Wide Web.

První verzí HTML bylo HTML 1.0, které se hodně lišilo od podoby, kterou v současnosti známe, především co se týkalo omezení. Protože internet v té době nebyl tolik populární, tak se jen málo lidí věnovalo webovému vývoji a i ti mohli vytvářet jen velmi jednoduché texty (původní HTML obsahovalo totiž pouze 22 tagů). V roce 1995 vyšla verze HTML 2.0, která však nepřinesla nijak výrazné změny. Od toho roku se však internet stával stále populárnější a lidé se začali zajímat i o HTML. Vznikly tak větší požadavky na schopnosti, více tagů a větší atraktivnost internetových stránek. Současně však mezi

internetovými prohlížeči dominoval Netscape Navigator, který rychle reagoval na požadavky a vyvinul své vlastní proprietární tagy, které však fungovaly pouze v jejich prohlížeči. Po marných pokusech aplikovat tyto nové možnosti i do jiných prohlížečů, byla představena nová verze HTML 3.0, která obsahovala několik nových schopností a mnohem větší příležitosti pro webové vývojáře. Bohužel se však ukázalo, že prohlížeče jsou s těmito novinkami příliš pomalé a tak bylo od tohoto návrhu opuštěno.

2.1.2 HTML 4.01

V prosinci 1997 byla vydána verze HTML 4.0 konsorciem W3C Recommendation, která přinesla mnoho změn. Verze nabízela tři různé varianty – Strict, Transitional a Frameset. O dva roky později vyšla upravená verze 4.01, která nabízela stejné varianty jako předchozí verze, ze které byly upraveny některé drobné chyby. V nové verzi vznikla možnost stylizování pomocí kaskádových stylů, které se pak od HTML postupně oddělily. Nová verze umožňovala vkládat kaskádové styly, skripty, objekty, komplexnější tabulky, komplexnější formuláře, zlepšila možnosti tisku a zdokonalila webovou přístupnost pro lidi s určitými poruchami.

Verze 4.01 se na dlouho stala stabilní verzí a konsorcium W3C po jejím vydání oznámilo, že se HTML už nebude vyvíjet. Budoucnost měla patřit XHTML, která však ale pouze použila značky z verze HTML 4.01 tak, aby to splňovalo požadavky standardů XML. Během vývoje se z XHTML nakonec stala slepá větev, která se již nadále nevyvíjela.

2.1.3 HTML5

Největší změny však přinesla verze HTML5, na které od roku 2004 pracuje pracovní skupina The Web Hypertext Application Technology Working Group, zkráceně WHATWG, a teprve v prosinci roku 2012 se verzi dostalo tzv. W3C Candidate Recommendation. Hlavním cílem verze bylo zdokonalit programovací jazyk tak, aby podporoval moderní multimédia a současně si zachoval jednoduchou čitelnost a konzistentnost pro různé aplikace, jako například internetové prohlížeče, parsery apod. Verze HTML5 v sobě obsahuje kromě HTML 4 také XHTML1, které definuje XML serializaci a DOM Level 2 HTML, které definuje JavaScriptové API pro HTML i pro XHTML. (van Kesteren, a další, 2011)

Dalším cílem této verze je definovat samostatný značkovací jazyk, který může být napsán jak v HTML tak i v XHTML syntaxi. To zahrnuje větší podporu rozšíření a zlepšení

značek pro dokumenty, aplikační programovací rozhraní (API) a pro komplexní webové aplikace. Díky těmto důvodům se z HTML5 stal potencionální kandidát pro mobilní aplikace. Mnoho funkcí HTML5 byly vytvořeny s ohledem na to, že budou pracovat na mobilních zařízeních, jako jsou například „smartphony“ nebo tablety.

HTML5 se stala populární především proto, že přinesla mnoho nových funkcí a výhod z nich plynoucí. S touto verzí můžeme například přehrávat multimédia v internetovém prohlížeči, vytvářet v něm aplikace, které mohou fungovat i bez internetového připojení nebo používat stále populárnější funkci Drag-and-drop. Od předešlé verze přinesla HTML5 možnosti integrovat škálovatelnou vektorovou grafiku (SVG) a využívat matematických vzorců ze specifikace MathML. Verze přidala několik nových elementů, které napomáhají lepší struktuře dokumentů jako například `<article>`, `<canvas>`, `<section>`, `<video>` atd. Do formulářových prvků, resp. do elementu `input` přibyly nové typy kontrol jako třeba zda se jedná o email, URL adresu nebo datum a čas. Dalším příspěvkem této verze jsou globální parametry (které mohou být aplikovány pro všechny elementy) - `id`, `tabindex`, `hidden` a `data-*`.

Specifikace má být oficiálně dle W3C schválena jako standard do konce roku 2014, nicméně v současnosti se již v nejpopulárnějších prohlížečích velmi často používá.

2.2 CSS

Kaskádové styly neboli CSS vznikly již roku 1997 z důvodu tvorby lepších stránek co se týkalo vzhledu. Webovým vývojářům CSS umožnilo oddělit obsah od vzhledu a tak mohl jazyk HTML dosáhnout svého skutečného cíle – značkovacího obsahu bez nutnosti starat se o vzhled.

Do roku 2000 nebyly kaskádové styly nijak zvlášť populární, protože internetové prohlížeče poskytovali z nabídky CSS akorát pouhou úpravu fontu nebo barvu stránek. Postupně se však styly začaly včleňovat do všech moderních prohlížečů a tím odstartovaly postupný vývoj, který v současnosti obsahuje tři verze: CSS Level 1, CSS Level 2 a nejnovější CSS Level 3, který je rozdělen do několika samostatných dokumentů, zvaných modulů. Každý modul přináší nějaké novinky nebo rozšiřuje možnosti CSS 2. Naopak díky této modulárnosti mají tyto moduly různé stability a stavy. V polovině roku 2012 publikovala společnost CSS Working Group., která má na starost vývoj kaskádových stylů, přes padesát CSS modulů. Avšak pouhé čtyři moduly byly organizací W3C předány jakožto doporučení

pro oficiální začlenění (jmenovitě to byly Media Queries, jmenné prostory, selektory a CSS barva, obsahující například transparentnost).

2.3 PHP

PHP: Hypertext preprocessor, zkráceně PHP, je serverový skriptovací jazyk, který umožňuje vytvářet dynamické webové stránky na straně serveru, kde se zpracovávají požadavky a následně se zpět posílají již výsledky jeho činnosti.

Vznik PHP se zrodil roku 1995, kdy si dánský programátor Rasmus Lerdorf vyvinul vlastní skript napsaný v jazyce Perl (později přepsán do jazyka C), který mu umožňoval zjistit, kolik návštěvníků četlo obsah jeho webových stránek (Gilmore, 2007). Tuto sadu nástrojů pojmenoval jako Personal Home Page. Sada byla natolik populární, že donutila Lerdorfa vytvářet další dodatky. Jeden z nich měl převádět data zadaná do formuláře HTML do symbolických proměnných, aby je uživatelé mohli exportovat na jiné systému (Gilmore, 2007). V roce 1997 pak vyšlo PHP 2.0 neboli Personal Home Page – Form Interpreter (PHP/FI). Následoval expanzivní růst a vývoj jazyka a podle měření internetové společnosti Netcraft z ledna letošního roku, využívalo PHP 244 milionů stránek, to znamená 39% všech stránek, které společnost monitoruje.

Jeho popularitu navíc zvyšují CMS systémy jako například WordPress, Joomla a Drupal, společně s několika dalšími e-commerce systémy jako Zencart, osCommerce a Magento.

I když má jazyk PHP několik významných konkurentů (např. ASP firmy Microsoft nebo JSP založený na jazyce Java), tak jsem si ho vybral právě proto, že je velmi populární a tudíž lze snadno najít řešení na požadovaný problém jak v literatuře, tak na internetu. Další pozitivní věcí byla relativní jednoduchost jazyka. Pro to, abych mohl napsat jednoduchý skript, jsem nemusel používat žádnou dodatečnou knihovnu a postačil mi jeden řádek kódu. Dalšími výhodami, díky kterým bylo rozhodnuto pro PHP, byly tyto:

- Není zde žádné omezení co se týče licence. PHP lze používat, stahovat, kopírovat apod. bez jakékoliv nutnosti platit poplatky či kontaktovat autora.
- PHP je multiplatformní, takže kód bude pracovat stejně ať už na Windows, Linux nebo na jiném operačním systému.

- Velmi kvalitní dokumentace. Veškeré funkce lze dohledat na oficiálních stránkách PHP a zjistit si, k čemu slouží, včetně jejich příkladů

2.4 MySQL

MySQL je světově nejpoužívanější relační databázový systém (neboli RDBMS), který je k dostání jak v komerční verzi (a nabízí další funkcionality), tak v rozšířenější verzi open source. MySQL je velmi rychlá databáze, která však na úkor výkonu musela odebrat některé funkcionality, jako například procedury, triggery nebo transakce. Nicméně se s každým vydáním posunuje blíže ke standardu SQL-92 (někdy uveden jako SQL2), jak uvádí Gilmore (2007).

Tento databázový systém byl vybrán především díky jeho důležitým aspektům:

- Výkon – již od počátku vývojáři MySQL kladli důraz na jeho rychlost, která je pro mou aplikaci klíčová
- Fulltextové vyhledávání – systém má poměrně dobře propracované fulltextové vyhledávání a tak ho lze jednoduše zrealizovat
- Cachování dotazů – MySQL si dokáže ukládat v cache dotazy včetně jejich výsledků. To rapidně napomáhá jeho rychlosti
- Velmi aktivní a rozsáhlá komunita uživatelů – díky tomu se systém rychle vyvíjí, často vycházejí nové verze, ve které jsou následně odhaleny chyby a jsou opraveny. Navíc existují nepřeberné množství publikací, fór a jiných prostředků, které umožňují najít řešení na konkrétní problém

2.5 JavaScript

JavaScript (zkráceně jako JS) je skriptovací jazyk, který se používá v internetových stránkách a umožňuje jim poskytovat funkcionalitu jako například dynamické nabídky, animace, efekty obrázků či jiný interaktivní design. JavaScript patří mezi skripty, které pracují na straně uživatele, i když se v poslední době intenzivně rozšiřují i JS systémy pracující na straně serveru (např. gcj, Node.js nebo Apache).

Standardizovaná verze JavaScriptu je pojmenována jako ECMAScript (která je podporována ve všech prohlížečích) a i když jazyk obsahuje slovo Java, tak s programovacím jazykem Java má společného jen pramálo a jejich sémantika se velmi liší. Syntaxe

JavaScriptu vychází z programovacího jazyka C, nicméně principy fungování převzal z programovacích jazyků Self a Scheme.

JavaScript je jazyk interpretovaný (tudíž se nemusí kompilovat), objektový, závislý na prohlížeči a case sensitivní (záleží na velikosti písmen). I když je JavaScript populární a hojně využívaný, má určité omezení, se kterými se musí při vývoji počítat. Je ho možné například uživatelem zakázat, neumí přistupovat k souborům a neumí žádná data uložit (výjimkou jsou cookies).

2.6 AJAX

AJAX, neboli Asynchronous JavaScript and XML, slouží k vyměňování si dat se serverem asynchronně (v pozadí) a aktualizací částí internetových stránek a to vše bez nutnosti opětovně načítat celou stránku. I přes jeho název, není nutné využívat XML (naopak nejčastěji se používá JSON) a požadavky nemusí být asynchronní.

Díky AJAXu lze plně využívat aktivitu serveru a současně provádět požadavky klienta, což dosud nebylo možné řešit. Podle Darieho (2006) „*AJAX přichází s řešením pro vyrovnání zátěže mezi klientem a serverem tak, že jim umožní komunikovat na pozadí, zatímco uživatel pracuje se stránkou*.“

Protože AJAX pro svou práci využívá JavaScript, je třeba při vývoji myslet na podobné úskalí. Je tedy možné ho na straně klienta vypnout (a tudíž bude AJAX nefunkční) a je potřeba mít místo toho náhradní řešení. Dále je zde problém s tlačítkem Zpět v internetových prohlížečích. Protože je stránka aktualizována bez nutnosti načítání stránky, může tlačítko Zpět navést uživatele na jeho poslední navštívenou stránku, která však může být pro něj nežádoucí. S tím také souvisí velmi těžké bookmarkování v prohlížečích. V neposlední řadě může být problém s webovými crawlery, kteří systematicky „projíždějí“ World Wide Web, typicky za účelem webového indexování. Protože většina crawlerů nevykonává JavaScriptový kód, měly by veřejně indexovatelné webové aplikace poskytovat alternativní prostředky pro přístup k obsahu, které by pomohly vyhledávačům aplikaci indexovat.

2.7 PHP Framework

PHP Framework je strukturovaná knihovna tříd, která nám rapidně pomáhá při vývoji webových aplikací. Pomocí frameworku ušetříme mnoho času a zredukujeme zbytečně se opakující kód (včetně toho nevyužitého).

V minulosti, v době PHP 4, se začaly postupně rozšiřovat tzv. „proto-frameworky“ (respektive CMS), které obsahovaly mnohem komplexnější systémy. Nicméně podpora objektového programování v PHP 4, která byla pro tyto systémy důležitá, příliš nepřála. I když existovaly robustní balíčkovací systémy, jako například PEAR, bylo stále velmi obtížné vytvořit komplexní webovou aplikaci. S příchodem PHP 5 a její hlavní předností, objektově-orientovaného programování, se začaly objevovat stále složitější webové aplikace. Společně s nimi vzniky i PHP frameworky, které měly urychlit a zjednodušit vývoj nových a větších aplikací.

Postupem času tak vznikaly více či méně populární frameworky, které se však vzájemně lišily a nenabízely vždy stejné možností. Nicméně aby byl framework hojně využíván, musí splňovat určité minimální požadavky:

- **Databázová podpora –** Ne každý framework podporuje všechny požadované databázové systémy a tak je nutné volit takový framework, který splňuje požadavky
- **Podpora komunity** – Framework by měl mít širokou komunitu, která by měla být aktivní a ochotna v případě potíží pomoci.
- **Podpora dokumentace –** Důležitým prvkem je i dobře a přehledně napsaná dokumentace. Ta by měla být aktualizovaná v případě změn.
- **MVC/MVP architektura –** Framework by měl využívat MVC/MVP architekturu z důvodů výhod, kterých z ní plyne. Dobrý framework by měl také obsahovat dodatečné knihovny, pluginy, nápovědy a možnost dalšího rozšíření.

Dalším důležitým prvkem pro výběr frameworku je dostatečná podpora ze strany vývojářů. Malý a nepřesvědčivý framework značí, že byl vytvořen individuálně s určitými nedostatky znalostí jazyka PHP. Dalším kritériem pro volbu, na kterou bychom neměli zapomenout, je zjištění si technických požadavků frameworku. Všeobecně platí, že PHP 4.3 je minimum a nejvhodnější je PHP 5.1.1 a vyšší verze. Co se týče databází, tak by měl framework podporovat MySQL verze 4.0, 4.1.x, 5.0, případně vyšší.

Mezi nejznámější PHP frameworky patří například Nette Framework, Zend Framework, CakePHP, CodeIgniter, Symphony, Kohana, Seagull nebo PRADO.

2.7.1 Architektura MVC

Hlavní myšlenkou všech populárnějších PHP frameworků je využití softwarové architektury Model View Controller, neboli zkráceně MVC. Jedná se o architektonický vzor v programování, který izoluje business logiku od uživatelského rozhraní, umožňující upravovat jednotlivé části odděleně. První originální návrh MVC formuloval Trygve Reenskaug v roce 1979 a jako první ho začal používat programovací jazyk Smalltalk-80.

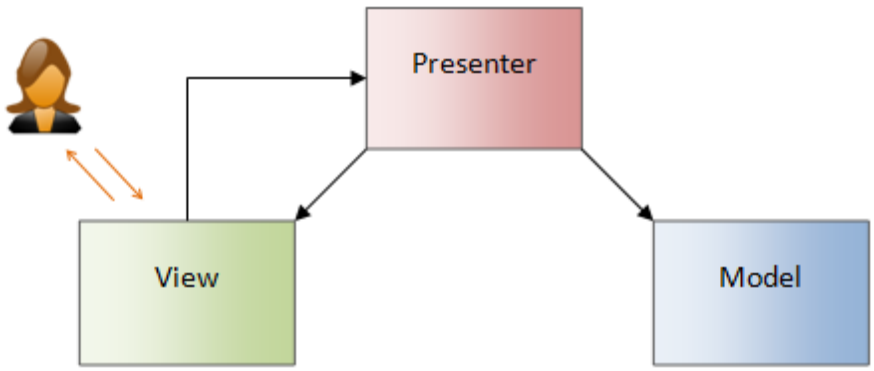
MVC se skládá z modelu, který představuje data a business logiku, pohledu (View), představující výstupní vrstvu pro uživatele (kde výstup může být HTML, XML nebo JSON) a z řadiče (Controller), který funguje jako prostředník a pracuje s aplikační logikou. V podstatě tak MVC odstraňuje mnoho nejasností a zjednodušuje celý vývojový proces. Je tak možné pracovat s individuální částí, zatímco ostatní zůstanou nedotčeny. V zásadě tím dělá kódování v PHP rychlejší a méně komplikované. Této architektury využívá např. Zend Framework.

2.7.2 Architektura MVP

Vzor Model View Presenter, zkráceně MVP, vychází z architektury MVC. V tomto vzoru View reaguje na události uživatelského rozhraní a v případě potřeby volá Controller, v této architektuře pojmenované jako Presenter, který obsahuje veškerou aplikační logiku a je schopen adekvátně odpovědět na požadavek. Presenter je tak zodpovědný za aktualizaci View s novými daty generované Modelem. View nyní plně kontroluje uživatelský výstup i vstup a téměř vždy pracuje pouze s Presenterem (na rozdíl od MVC), takže i tato vazba je silnější než v případě vzoru MVC. (Bernard, 2009)

Architekturu MVP pak ještě rozvinul Martin Fowler, který ji rozdělil na dvě variace:

1. Vzor Supervising Controller – Vrstva View má zodpovědnost za zobrazení dat z modelu a jakákoliv složitější prezentační logika je potom přesunuta do Presenteru, který má dovoleno s View libovolně manipulovat. (Bernard, 2009)
2. Vzor Passive View – Jak vyplývá z obrázku č. 1, View zde nemá téměř žádnou zodpovědnost a vše musí obstarat Presenter, tedy i případnou vazbu mezi Modelem a View. Tento typ vzoru využívá například Nette Framework.



Obrázek č. 1 - Vzor Passive View architektury MVP

2.8 Nette Framework

Nette Framework je účinný nástroj napsaný v PHP 5 využívajíce plného využití objektově orientovaného programování. Vývoj započal roku 2004 a o čtyři roky později byla vydána první verze 0.7. Za jeho vývojem stojí zkušený český PHP vývojář David Grudl, který vytvořil neméně známé aplikace Texy! a databázovou vrstvu Dibi, kterou dříve Nette využíval. Jeho licence vychází z BSD, která patří k těm nejsvobodnějším. *Nette je koncipován jako „otevřený" a lze ho kombinovat s jiným otevřeným frameworkem, jako je například Zend Framework*. (Grudl, 2009)

Protože se jedná o český produkt, tak má také silnou a aktivní komunitu, která je jedna z nejaktivnějších komunit českých PHP vývojářů. Nicméně postupně se rozšiřuje i zahraniční komunita, která je však výrazně nižší. Nette také používají významné tuzemské společnosti, jako jsou například Internet Info, s.r.o., Mladá fronta, Slevomat a mnoho dalších. (Grudl, 2009) A podle testu uveřejněného na serveru Root.cz je jedním z nejvýkonnějších frameworku.

Nette využívá kvalitní vzorový návrh MVP, který je obdobou vzoru MVC. Zaměřuje se především na snížení všech bezpečnostních rizik a znovu použitelnost kódu. Podstatným rysem je také náhled na webovou stránku jako soustavu komponent.

2.8.1 Výhody Nette Framework

Na rozdíl od jiných frameworku, poskytuje Nette některé věci, které jsou považovány za unikátní a jedny z hlavních předností, proč byl tento framework vybrán pro vývoj aplikace:

- **Ladění –** Nette obsahuje utilitu nazvanou jako „laděnka“, která programátorovi zobrazí chybovou zprávu s částí zdrojového kódu, včetně zvýrazněného řádku, kde k chybě došlo. Navíc se chyba automaticky loguje a je možné si ji nechat automaticky poslat na náš email. Protože se jedná o nástroj, který je především potřeba ve vývojovém prostředí, tak v produkčním prostředí je laděnka explicitně vypnutá, ačkoliv je možné ladící nástroj „natvrdo“ zapnout (i když se to velmi nedoporučuje).
- **Minimalizovaná verze –** k dispozici je možné využít i minimalizovanou podobu Nette, která vznikla sloučením všech souborů a odstranění komentářů a mezer. Verze je funkčně rovnocenná a je určena především pro produkční nasazení (snadnější upload, vyšší rychlost).
- **Autoloading –** Nette využívá komponentu RobotLoader, která nám umožňuje využívat (nahrávat) libovolné soubory nezávisle na adresářové struktuře. Tyto soubory pak automaticky indexuje a uchovává v cache.
- **Obousměrné routování mezi URL a akcí presenteru –** díky tomu je možné z URL odvodit akci presenteru a naopak. Lze tak strukturu URL navrhovat nezávisle na zbytku aplikace. Navíc je možné relativně snadnou používat tzv. „Cool URL“.
- **Práce s formuláři –** Nette výrazně usnadňuje tvorbu a zpracování formulářů v aplikacích. Také bere ohled na bezpečnost vstupů a chrání před útoky typu Cross Site Scripting, Cross-Site Request Forgery nebo UTF-8 attack. Umožňuje nám to tak vyhnout se zbytečnému psaní dvojí validace (na straně serveru a JavaScriptu).

2.8.2 Nevýhody Nette Framework

Ačkoliv je Nette Framework vhodný pro začátečníky, je rychlý a poskytuje mnoho výhod, tak stejně jako ostatní frameworky, i Nette má své nevýhody, které ztěžují práci s ním:

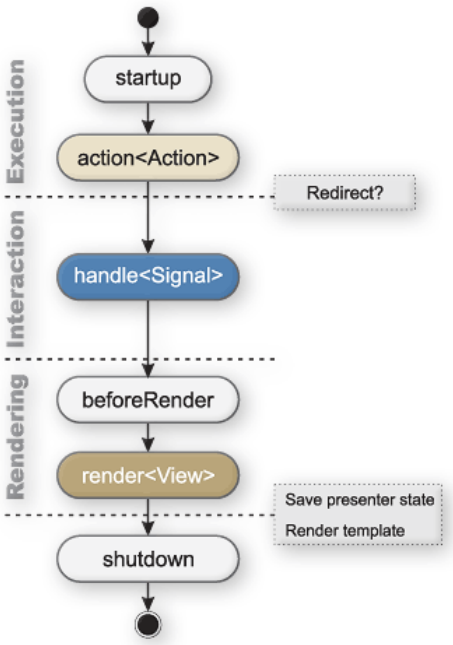
- **Velmi nízká mezinárodní účast** - i když Nette vznikl již v roce 2008 a vznikla okolo něj velmi rozšířená základna českých uživatelů, v zahraničí je tento framework stále velkou neznámou. Důvodem je také absence úplné anglické dokumentace, kde mnoho částí, zejména určených pro začátečníky, není přeloženo nebo zcela chybí.

- **Nejistá budoucnost** - směr a vývoj frameworku určuje především jeden člověk, a to jeho autor David Grudl. V současné době je za vývoj zodpovědná nadace Nette Foundation, nicméně hlavním „tahounem“ projektu je stále Grudl a není tak jisté, jak by se Nette Framework vyvíjel v případě, že by ho opustil.
- **Rozdílné návody** - pro začátečníky existují na oficiálních stránkách Nette návody, které mají usnadnit pochopení frameworku. Nicméně tyto návody nejsou dotaženy do konce a některé pasáže jsou chybné a začátečník je tak pouze odkázán na fórum. Tyto návody se navíc odlišují od těch, které se nacházejí na GitHubu a na které návody odkazují.
- **Špatná kompatibilita verzí** - první verzí bylo Nette 0.7, které se postupně vyvíjelo do verze 0.9, která delší dobu zůstala jako stabilní. O něco později vznikla rovnou verze 2.0, která však měla některé prvky odlišné od starší verze. Tím vznikla nekompatibilita těchto dvou verzí a vývojáři při přechodu museli řešit mnoho problémů.
- **Příliš rychlý vývoj** - Nette Framework se velmi rychle vyvíjí a spolu s ním i dokumentace. To, podle čeho se začátečník řídil, se může za půl roku velmi změnit. Příkladem může být začleňování moderního postupu tzv. „Dependency Injection“, kdy se třída přímo hlásí ke svým závislostem, které jsou pak dále „injektovány“. S rychlým vývojem souvisí i absence knižních publikací o Nette.

2.8.3 Architektura systému Nette

Presenter

V presenteru se odehrává veškerá logika aplikace. Přijímá požadavky klienta z pohledu a v případě potřeby komunikuje s potřebným modelem. Presenter se řídí svým životním cyklem, během kterého můžeme rozhodovat, jak bude aplikace probíhat. Jak znázorňuje obrázek č. 2., cyklus se skládá celkem ze šesti fází.



Obrázek č. 2 - Životní cyklus presenteru

- **startup() –** zde dochází k inicializaci presenteru. Většinou zde registrujeme všechny služby, které jsme definovali v systémovém DI kontejneru a které budeme potřebovat.
- **action<NazevAkce>() –** zde se provádí tzv. „akce presenteru“. Protože tato fáze probíhá ještě před samotným vykreslením šablony, můžeme zde provádět akce, podle kterých se dále můžeme rozhodnout o případném přesměrování na jiný pohled. Jedná se vlastně o podobnou fázi jako render<View>, nicméně se zde nic nevykresluje.
- **handle<NazevSignalu>() –** v této fázi se nám zpracovávají přijaté signály, neboli subrequesty, které můžeme obdržet od různých komponent nebo z AJAXových požadavků. Protože se tato metoda volá nad vykreslením šablony, vykonávají se akce, aniž by se změnil pohled.
- **beforeRender() –** těsně před samotným vykreslením šablony můžeme například měnit nastavení šablony, předávat proměnné společné pro více šablony apod.
- **render<NazevView>() –** zde se jedná o typické vykreslení. Obvykle se zde předávají proměnné do šablony, které se pak budou používat.
- **shutdown() –** metoda zahájí ukončení životního cyklu presenteru. Tato metoda se běžně explicitně neuvádí a je provedena automaticky.

Společným předkem všech presenterů v aplikaci je pak abstraktní třída `BasePresenter`, která dědí základní třídu `Presenter`. V tomto společném předku provádíme akce, které jsou společné pro všechny presentery jej dědící, jako jsou například registrování služeb či inicializace dat. V naší webové aplikaci zde provádíme akce, které jsou klientovi neustále k dispozici, jako je například menu, navigace, vyhledávání produktů nebo přihlašování, respektive odhlašování.

Model

Modely jsou jedním z klíčových prvků webové aplikace. Modely jsou nezávislé na prezentační logice, tedy té části aplikace, která model prezentuje uživateli a zpětně překládá jeho požadavky. (Procházka, 2011) Jednotlivé modelové třídy reprezentují entity, které jsou odrazem reálného světa. V naší aplikaci existují tedy modely jako například `BasketModel`, `CategoryModel`, `OrderModel` nebo `ProductModel`. Každý model reprezentuje jednu, maximálně dvě, tabulky v databázi. S touto tabulkou pak model pracuje, vytváří záznamy, edituje nebo je maže.

Předkem všech modelů je ve webové aplikace abstraktní třída `TableModel`. Jejím hlavním úkolem je navázat připojení k databázi a případě chyby vyvolat výjimku. Dále obsahuje metody, které jsou společné pro více modelů, jež je mohou dědit či přepsat.

Zvláštním modelem je pak `Authenticator`, který je již dodáván v Nette distribuci. Jeho účelem je autentizace uživatelů při jejich přihlášení. Jako jediný model implementuje rozhraní `IAuthenticator`, který obsahuje konstanty popisující chybu při přihlášení. Pro naši aplikaci je tento model velmi důležitý, protože zde vytváříme hash a ověřujeme heslo pomocí algoritmu Bcrypt.

Pohled

Poslední neméně důležitou částí jsou pohledy, v Nette zastoupeny jako Latte šablony. Pohled a Presenter jsou velmi úzce spojeni a ve většině případů je při vytvoření presenteru vytvořena také odpovídající šablona. Aby během životního cyklu presenteru byla správně naplněna zvolená šablona (fáze `render<NazevView>`), je nutné dodržovat umístění šablony. Ta musí být uložena v adresáři pojmenovaném stejně jako Presenter, který renderovací fázi, respektive vykreslení požadované šablony, volá.

Předkem všech šablon je soubor `@layout.latte`, který se využívá pro vykreslení těch částí webové aplikace, které chceme zobrazit při každém HTTP požadavku. Do tohoto souboru také vkládáme všechny odkazy na externí JavaScripty, kaskádové styly a další inicializační věci.

Za velmi užitečnou vlastnost šablon považuji jejich dědičnost, a to pomocí bloků. Šablony můžeme zanořovat do dalších šablon nebo je naopak rozšiřovat o svůj obsah a vytvářet tak určitou hierarchii.

Když provádíme jakýkoliv vstup, je třeba zajistit zamezení všech bezpečnostních děr ve skriptech. Na to je potřeba neustále myslet a tak například při každém volání obyčejné funkce jako je `echo($promenna)`, musíme zajistit správné escapování (převést znaky na jiné známé sekvence). Tím pokaždé získáme kód ve tvaru `echo htmlspecialchars($promenna, ENT_QUOTES)`. Jak už z příkladu vyplývá, jedná se o velmi zdlouhavou práci, na kterou však vývojář nesmí zapomenout. Nette Framework však nabízí relativně jednoduché řešení, jak si zajistit ochranu proti útokům typu XSS. Latte šablony totiž disponují automaticky zaregistrovaným Latte filtrem, který proměnné automaticky escapuje. V případě, že by programátor nechtěl, abych se proměnná automaticky escapovala, stačí jen před proměnnou vložit vykřičník, například takto `{!$promenna}`.

Za další výhodu šablonovacího systému Latte je považována existence tzv. maker, které slouží pro jednodušší zápis funkcí v šabloně. Nette obsahuje řadu maker, které můžeme jednoduše použít a ulehčit si tak mnoho starostí. Mezi ně patří například výpis proměnných a výrazů, podmínky, cykly, AJAX makra nebo makra pro ladění webové aplikace.

Dalším významným nástrojem šablonovacího systému je využití funkcí, tzv. „helperů“, kteří pomáhají formátovat a upravovat data do požadovaného výsledku. Nette disponuje řadou helperů, kteří se velmi snadno používají. Pro jejich zápis stačí helpery uvést za svislou čárou, lze je zřetězit a aplikují se v pořadí od levého k pravému. Parametry se zadávají oddělené dvojtečkou nebo čárkou. (Grudl, 2011)

2.8.4 Routování

V moderních dynamických aplikacích je potřeba sbírat požadavky zvenčí a podle nich adekvátně reagovat a presentovat správné data. Typickým příkladem pro požadavek zvenčí je URL adresa, podle které se generuje patřičná akce. Abychom stanovili určitá pravidla chování aplikace, využívá se tzv. routování.

Nette Framework používá obousměrné překládání mezi URL a akcí presenteru, díky kterému lze parsovat i generovat požadované adresy. K routování lze přidat i masky, které pak jednoduchým způsobem definují, jak mají adresy vypadat. Společně s nimi můžeme tak vytvořit tzv. „user-friendly URL“, které již neobsahují query stringy a namísto toho obsahují pouze cestu k požadovanému zdroji. Tato technika vytváření URL adres se často používá pro lepší estetiku, použitelnost nebo pro lepší optimalizaci vyhledávačů (tzv. SEO). Dalšími důvody tvorby „user-friendly URL“ pro webové stránky a aplikace je větší trvanlivost URL adresy, která dělá z World Wide Web více stabilní a použitelnější systém. (Berners-Lee, 1998) Navíc jsou tyto adresy „lidštější“ a lépe zapamatovatelné.

2.9 Zend Framework

Zend Framework patří mezi nejznámější objektově-orientovaný framework napsaný v PHP 5. Je jednoduchý, práce s ním není těžká a má obrovskou základnu aktivních uživatelů a vývojářů. Obsahuje robustní množství funkcí a rozšíření a je určen především pro komerční produkty, jako například komerční webový server. Využívá vzorový návrh MVC.

Zend je podobný frameworku CodeIgniter a CakePHP, ale jeho velmi těžko čitelná dokumentace založená na návrhových vzorech z něj nedělá vhodného kandidáta pro začátečníky a je potřeba mít velké znalosti jazyka PHP.

Za jeho vývojem stojí vývojáři jádra PHP a tak má velmi pevné základy, které jsou určitě výhodou. Zend je pravděpodobně nevyhledávanější framework v rámci jazyka PHP a poskytuje programátorům výhody jako například při vytváření formulářů, filtrování dat nebo internacionalizaci. Zend také umožňuje využít širokou podporu a nabízí školení či certifikaci ohledně jeho produktů.

2.10 Pomocné knihovny

Knihovny obsahují předem vytvořené funkce a procedury, které mají za úkol pomáhat a ulehčovat programátorovi vývoj aplikace.

2.10.1 NotORM

Object Relation Mapping, zkráceně ORM, je abstraktní vrstva, která slouží při vytváření aplikací, kdy objektový model používá množinu business tříd – tříd, jejichž objekty

reprezentují záznamy uložené v databázi (Lecky-Thompson, a další, 2010). Výsledkem pak může být například systém Object-Oriented Database Management Systém (zkráceně OODMS), který zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem a tudíž lze časté databázové dotazy (například vytvoření, čtení, aktualizace nebo odstranění) provádět bez nutnosti psát jakýkoliv kód v jazyce SQL.

Nicméně tyto databázové knihovny jsou obvykle velmi robustní (například populární Doctrine zabírá až 2MB) a nutí programátora naučit se jejich kompletní manuál, který může být někdy velmi obtížný.

Naproti tomu je zde NotORM napsané Jakubem Vránou, PHP knihovna která slouží pro manipulaci s databází. Je velmi lehká na naučení, není potřeba vytvářet třídu pro každou tabulku v databázi a zvládá tabulkové vztahy často rychleji než samotné nativní PDO.

I když je NotORM implementované v Nette Frameworku, nejedná se o kompletní převzetí knihovny. Autor knihovny Jakub Vrána (2011) píše, že se „*jedná spíše o zařazení odvozené práce s tím, že Nette zatím stále nepodporuje některé obraty (např. definici vlastní konvence pojmenování sloupců) a využívá odlišnou syntaxi přístupu k tabulkám a sloupcům*“.

2.10.2 Latte

Latte je šablonovací systém, který je součástí Nette Frameworku. Umožňuje zobrazení proměnných, využívat cykly, podmínky či průchod polí (včetně vícerozměrných). Tyto funkce jsou v Latte nazývány jako makra a zapisují se pomocí dvou druhů speciálních značek

- makra jsou ve složených závorkách, například {foreach…}
- n:makra, například n:href=“…“

Dále jsou v Latte k dispozici tzv. „helpery“, které programátorům umožňují upravit nebo formátovat data do požadované podoby, aniž by bylo třeba využívat některé speciální funkce PHP.

Protože překládá šablony do nativního PHP kódu a ukládá do cashe, je Latte velmi rychlé. Hlavní výhodou systému je však ušetření mnoho času se zajištěním bezpečnosti a zranitelností například před nejčastějšími útoky typu XSS (Latte totiž vypisované proměnné automaticky escapuje pomocí technologie Context-Aware Escaping, která rozezná, ve které části dokumentu se makro nachází a podle toho zvolí správné escapování (Grudl, 2011)).

2.10.3 jQuery

jQuery je multiplatformní JavaScriptová knihovna, která je navržena pro jednodušší skripty v HTML. Vznikla v roce 2006 a jedná se v současnosti o nejpopulárnější JavaScriptovou knihovnu.

jQuery je open-source software a jeho syntaxe je navržena k jednoduché navigaci v dokumentu, výběru DOM elementů, vytváření animací, vytváření událostí a k vývoji AJAX aplikací. Knihovna také umožňuje vývojářům přidávat další své vlastní pluginy a rozšířit tak její funkcionalitu. V současnosti existuje přes 700 pluginů, které pokrývají velkou škálu funkcionalit, jako jsou například AJAXové nástroje, webové služby, datagridy, dynamické seznamy, XML a XSLT nástroje, funkce drag and drop, tvorba událostí, práce s cookies nebo modální okna.

Modulárnost tak zajišťuje tvorbu interaktivních dynamických internetových stránek a aplikací.

2.10.4 TinyMCE

TinyMCE je vizuální editor WYSIWYG, který slouží pro zadávání textů do webových formulářů. Jeho hlavní schopností je konvertování HTML a jeho snadná integrace s CMS systémy, jako jsou například Drupal, Joomla! nebo WordPress.

Editor nabízí formátovací nástroj, který umožňuje měnit kurzívu, tučnost, font, přidávat odstavce nebo videa a obrázky. TinyMCE je kompatibilní s většinou prohlížečů a operačních systémů. Obsahuje mnoho pluginů, mezi nimiž je i možnost přidání českého jazyka do editoru.

2.10.5 Twitter Bootstrap

Twitter Bootstrap je kolekce nástrojů, které umožňují vytvářet internetové stránky a aplikace. Zahrnují šablony HTML a CSS, díky kterým můžeme využívat jednolité typografie, tlačítka, formuláře, grafy, navigace a další komponenty včetně JavaScriptového rozšíření.

Bootstrap vznikl ve společnosti Twitter jako framework, který měl zajistit konzistentnost mezi dalšími nástroji. Jeho popud na vznik vzešel především z využívání různých knihoven během vývoje Twitteru, což vedlo k častým nesrovnalostem a vysokou náročnost z hlediska údržby.

Bootstrap obsahuje nekompletní podporu pro HTML 5 a CSS 3, nicméně je kompatibilní se všemi hlavními prohlížeči. Základní funkčnost internetových stránek nebo aplikací je však možná pro všechny zařízení a prohlížeče. Od verze 2.0 také podporuje responzivní web design, který zaručí, že zobrazení stránky bude optimalizováno pro všechny druhy nejrůznějších zařízení (mobily, notebooky, netbooky, tablety atd.).

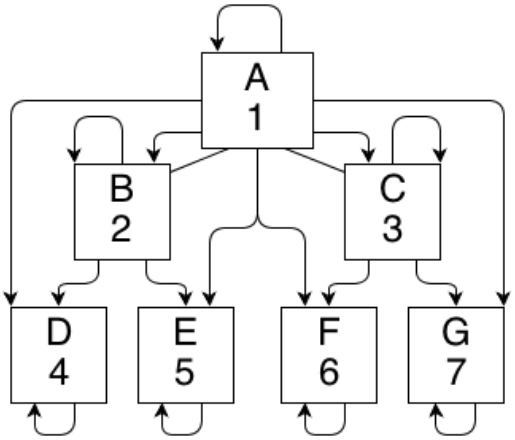
2.11 Hierarchická struktura dat

V internetové aplikaci je potřeba přidávat kategorie, jejich podkategorie a tak dále. K tomuto dosažení, je potřeba vytvořit určitou hierarchii, která bude organizována do stromové struktury dat. Protože se v aplikaci používá relační databáze, je dosažení této struktury nelehkým úkolem, na rozdíl například od objektových databází. Podle Zelenky „*v objektové databázi mohou být stromová data uložena přímo v takové podobě, jakou využívá aplikace, která se k této databázi připojuje. Naopak při použití relační databáze musíme data transformovat tak, abychom je mohli uložit do ploché relační tabulky, a při čtení dat z databáze je musíme zpětně transformovat do podoby stromu.*“ (2005) Aby bylo možné ukládat stromové data do relační databáze, existuje několik nejznámějších přístupů.

2.11.1 Closure Table

Metodu Closure Table poprvé navrhl softwarový profesionál Bill Karwin, který je také jejím největším propagátorem. V této hierarchické struktuře jsou uloženy všechny cesty stromu, nejen přímé reference rodič - potomek. Ukládá se také cesta s nulovou délkou, což znamená, že je uzel vlastním rodičem.

Jak z obrázku č. 3 vyplývá, pokud je uzel A rodičem uzlu B, B rodičem uzlu C, který je rodičem uzlu D, pak je potřeba uložit následující cestu: A-A, A-B, A-C, A-D, B-B, B-C, B-D, C-C, C-D, D-D. Díky tomuto systému je lehké vytvořit dotaz pro všechny potomky uzlu A nebo všechny předky uzlu D.



Obrázek č. 3 - Hierarchická struktura dat metodou Closure Table

Abychom mohli metodu používat, je třeba vytvořit v databázi novou tabulku, která bude ve vztahu M:N. Ta bude obsahovat celkem tři sloupce - ancestor, descendant a depth, který označuje hloubku zanoření pro daného rodiče.

Díky tomuto můžeme jediným jednoduchým SQL dotazem získat celý strom, například od kořene A:

```
SELECT *
FROM category c JOIN category_closure cc ON c.category_id = cc.descendant
WHERE ancestor = 1
```

Nebo obráceně, pokud chceme vypsat rodiče například uzlu D, stačí opět pouze jediný dotaz:

```
SELECT *
FROM category c JOIN category_closure cc ON (c.category_id = cc.ancestor)
WHERE cc.descendant = 4 AND cc.depth > 0
```

V této struktuře dat se ukládá relativně mnoho dat, kde jejich počet může být až n^2 záznamů. Podle Voráčka ale tato situace „*nastane pouze v případě, kdy ze stromu vyrobíme jakýsi lineární seznam, takže každý prvek (kromě posledního) má právě jednoho potomka.*“ (2012) Samotný autor této metody Bill Karwin uvádí, že v praxi je tento počet o mnoho menší a není tak nutné řešit paměťovou náročnost databáze. (Karwin, 2010)

2.12 Pomocné nástroje

2.12.1 Git

Při vývoji je často programováno ve smyslu pokus - omyl. To znamená upravení několika tříd a v případě, že vše funguje, je možné pokračovat ve vývoji. Problém však nastane, když bude po nějakém čase zjištěno, že se vývoj nachází ve slepé uličce a je třeba vše, na čem bylo v ten daný čas pracováno, zahodit. To by znamenalo znovu „projíždět“ všechny třídy, které byly editovány, a vrátit je zpět do původní podoby. Avšak v případě, kdy by už se jednalo o změny většího rozsahu, nebylo by možné uvědomit si rozsah změn.

Aby bylo možné se tomuto zbytečnému a zdlouhavému procesu vyhnout, bude využit distribuovaný systém správy verzí Git. Jedná se o velmi rozšířený verzovací systém, který obsahuje velmi širokou funkcionalitu. Na druhou stranu však díky těmto širokým možnostem je těžké naučit se ho a používat. Git byl vytvořen Linusem Torvaldsem, známým především zahájením vývoje jádra operačního systému Linux, který měl původně sloužit pro vývoj jádra Linuxu. Git umožňuje vést historii revizí nebo vytvářet nové větve (branche), které lze v pozdějším stádiu vývoje sloučit (merge), nebo zcela zahodit a vrátit tak celou aplikaci do původní podoby před vznikem dané větve.

Git sám o sobě nepotřebuje pro své fungování žádný server a celý repozitář lze tak mít u sebe na disku. V případě, ale že bychom chtěli spolupracovat s více vývojáři, či jim nabídnout části zdrojového kódu k nahlédnutí, potřebujeme použít některý server, který využívá souborový systém. I když takových serveru existuje mnoho, například Bitbucket, Google Code nebo SourceForge, byl vybrán nejznámější server pro hosting open-source projektů verzovaných v Gitu a to GitHub. Vybrán byl pro jeho přehlednost, jednoduchost na ovládání a navíc obsahoval mnoho funkcí.

2.13 Použité technologie modelování

2.13.1 Diagram případů užití

Diagram případů užití (anglicky Use Case) je v softwarovém a systémovém inženýrství termín, který popisuje, jak uživatel používá systém k dosažení určitého cíle. Diagram případů užití využívá techniky softwarového modelování, která definuje funkce, které mají být implementovány a řešení případných chyb, které mohou nastat.

Diagram případů užití definuje vztahy mezi externími aktéry a systémem. Diagram obsahuje tři základní elementy:

- Aktéři – typ uživatelů, kteří pracují se systémem. Může se jednat například o administrátora, uživatele, externí zařízení apod.
- Systém – diagramy zachycují funkční požadavky, které určují zamýšlené chování systému.
- Cíle – diagramy případů užití jsou obvykle iniciovány uživatelem za účelem splnění cíle a popisují aktivity a varianty spojené s dosažením tohoto cíle.

Případy užití jsou v jazyce UML zastoupeny ve tvaru oválu obsahující název případu užití. Aktéři jsou pak nejčastěji zobrazeni jako postavy zobrazující dané osoby. Jejich účast v systému je reprezentovaná pomocí čáry mezi aktérem a případu užití. Rámeček okolo případů užití představují hranice, které stanovují, co patří do vnitřní části systému a co do vnější části. Charakteristiky spojené s diagramy případů užití jsou tyto:

- Zorganizování funkčních požadavků
- Modelování cílů systému pomocí uživatelských interakcí
- Vytvoření scénáře užití od spouštěcích událostí až po konečné cíle
- Popsat základní akce a výjimečně tok událostí
- Zobrazení uživatelských povolení k funkčnosti jiné akce

2.13.2 Diagram aktivit

Diagramy aktivit se běžně používají pro podnikové modelování procesů, pro modelování logiky zachycené v diagramu případů užití nebo pro modelování podrobné logiky obchodních pravidel. Jedná se o grafické znázornění pracovních procesů (aktivit) a akcí, které podporují rozhodování, iteraci a souběžnost. Diagram nám tak prezentuje celkový řídící tok. Diagram aktivit jakožto objektově orientované UML je ekvivalentem diagramu datových toků (DFD).

Na rozdíl od použití diagramu případů užití, je v diagramu aktivit zřejmé, zda aktéři mohou provádět činnosti společně nebo nezávisle na sobě. Vzhledem k tomu, že je možné, aby diagram explicitně mohl popsat paralelní události, tak se diagram aktivit dobře hodí pro

ilustraci podnikových procesů, protože pracovní procesy se vyskytují jen vzácně lineárním způsobem, spíše naopak často vykazují paralelismus.

Diagramy aktivity jsou pak tvořeny několika základními elementy:

- **Inicializace aktivity** – v diagramu je zobrazena jako kruhovitý útvar vyplněný černou barvou. Jedná se o fázi, kdy je tok aktivity odstartován a ačkoliv není inicializační bod nutný, tak se vyznačuje především snadnější pochopení diagramu aktivit.
- **Akce –** jedná se o individuální krok v diagramu, který je prezentován jako zaoblený čtyřúhelník. Akce mohou obsahovat další vnořené aktivity včetně dalších akcí.
- **Řídící tok –** někdy označován jako hrana, spojuje jednotlivé komponenty v diagramu aktivit a je reprezentován jako šipka. Její směr určuje, v jakém pořadí se akce vykonají. Řídící toky mohou obsahovat i podmínky a tudíž proces může přejít na další akci pouze tehdy, pokud je korektně splněna i zadaná podmínka.
- **Rozhodnutí –** jedná se o část systému, kdy do jednoho bodu vstupuje jeden řídící tok a vystupují z něj dva a více. V diagramu je označen jako kosočtverec a jednotlivé podmínky pro pokračování procesu jsou dány v hranatých závorkách.
- **Spojení –** v diagramu označován stejným symbolem jako pro rozhodování – kosočtvercem. Jeho účelem je spojení všech příchozích toků. Vstup toků nemusí být synchronní a tak výstup, který je již jen jeden, čeká na „příchod“ všech toků.
- **Rozdělovník –** černá silná vertikální nebo horizontální čára v diagramu aktivit značí rozdělovník. Vstupuje do něj jeden tok a vystupuje z něj několik toků. Tímto elementem se označuje začátek paralelně prováděných akcí.
- **Spojovník –** je opakem rozdělovníku, kdy do něj vstupuje několik toků a výstupem pak je pouze jediný tok. V tomto elementu diagramu aktivit dochází k synchronizaci toků, a tudíž proces pokračuje až po „příchodu“ všech ostatních toků.
- **Příchozí událost –** v této části systému akce čeká, dokud se neprovede určitá událost. Teprve poté je tok, přicházející z této události, vykonán. Tento element je pro diagram aktivit důležitý, protož business procesy často reaguje na nějaké události.
- **Ukončení aktivity –** tento element značí, že aktivita byla dokončena. V diagramu je reprezentován jako černá tečka s ohraničením. Aktivita může obsahovat více

ukončovacích bodů. V případě paralelních větví, může k jejich skončení dojít kdykoliv a značí se koncem toku.

2.13.3 E-R diagram

E-R diagramy, zkráceně ERD, poskytují cenný nástroj pro modelování vztahů mezi databázovými subjekty v jasném a přesném formátu. Tento standardní přístup využívá řadu tvarů a linií popisujících strukturu databáze způsobem srozumitelným pro všechny druhy databází. Některé balíčky profesionálních aplikací, jako například Microsoft Access, MS SQL Server nebo Oracle, umožňují automaticky vytvářet E-R modely z již existujících databází.

E-R diagramy, popsané poprvé roku 1976 Petrem Chenem, tvoří tři nejdůležitější prvky:

- **Entita –** může být definovaná jako objekt, který je nezávislý na ostatních objektech a může být jednoznačně identifikovatelný. Entita je odrazem objektů reálného světa, v našem případě například jako produkt, objednávka, uživatel nebo kategorie. Entity se prezentují jako obdélníky a jejich jména jsou označovány podstatnými jmény.
- **Vztah –** síla E-R diagramu spočívá v jeho schopnosti přesně zobrazit informace o logických vztazích mezi entitami. Vztah nám tak určuje kolik entit je asociovaných v jednom vztahu (stupeň), počet výskytů entit účastnících se jednoho výskytu vztahu (kardinalita) a zda je „*vztah povinný či volitelný ze strany jedné či druhé entity, tedy zda každému výskytu musí nebo může odpovídat jeden nebo několik výskytů dané entity.*“ (Kaluža, a další, 2011)
- **Atribut –** nám určuje vlastnost entity nebo vztahu. Pokud se nejedná o slabou entitu, tak musí existovat alespoň jeden atribut, který je tzv. primární klíč a slouží především jako jednoznačný identifikátor entity. Většina atributů patří k jednoduchým atributům, které obsahuje atomické hodnoty, které nelze již dále rozložit. Na druhou stranu existují i složené atributy, které mají společný význam nebo použití.

2.14 Bezpečnost

Bezpečnost internetového obchodu je jedna z nejdůležitějších částí. Závisí na ní jak ochrana uživatelských dat, tak také image e-shopu, protože důvěra zákazníka může být klíčový aspekt, proč by se měl vrátit.

Nette Framework s touto důležitou částí velmi pomáhá, protože obsahuje mnoho možností, jak se proti útočníkům bránit a jak eliminovat výskyt bezpečnostních děr. Pomocí něj není potřeba řešit útoky typu:

- Cross-Site Scripting (XSS) - tento typ útoku je založen na jednoduchém principu, kdy útočník využije bezpečnostní díry a do stránky podstrčí vlastní kód, který má za následek narušení vzhledu stránky nebo ještě hůře získat data o uživatelích. Proti tomu se Nette Framework brání pomocí technologie Context-Aware Escaping, která všechny výstupy automaticky ošetřuje.
- Cross-Site Request Forgery (CSRF) - útok spočívá v tom, že využije uživatele navštívit jinou stránku, která provede určitou akci, aniž by o tom uživatel nějak věděl. Útočník musí znát webovou aplikaci a připravit si proto formulář s akcí, která se provede, až oběť navštíví útočníkovu stránku. Zároveň musí být uživatel přihlášen na stránce, která má být napadena. Proti tomu se lze v Nette bránit pouhým jediným příkazem při tvorbě formuláře `$form->addProtection();`. Takto jsou chráněny všechny formuláře v administrační část e-shopu.
- Session hijacking - jakmile se uživatel do webové aplikace přihlásí, vygeneruje server vlastní unikátní identifikátor, tzv. SESSION_ID, který zašle zpět ve formě cookies. Tento identifikátor si pak server přidělí danému uživateli a do té doby, než se odhlásí, ho považuje za autorizovaného a povolí mu přístup na jeho konto už bez dalšího zadávání údajů. Při odhlášení se SESSION_ID automaticky na straně vymaže a při opětovném přihlášení je vygenerován nový identifikátor. Útočník tak využívá tohoto sezení, kdy se snaží získat nebo podstrčit uživateli své session ID a díky tomu získá přístup do webové aplikace, aniž by znal heslo uživatele. Nette Framework však nakonfiguruje PHP automaticky a tak tomuto typu útoku brání.

Mezi důležité zabezpečení patří také ochrana hesla při registraci a přihlašování. To bude zajištěno pomocí komponenty Bcript. Jeho účelem je zjednodušeně z hesla vytvořit hash, který se uloží do databáze.

2.14.1 HTTPS

HTTPS je nástavbou protokolu HTTP a umožňuje zajistit šifrovanou komunikaci. Data jsou pak nejčastěji šifrovaná protokolem SSL nebo TLS. Díky tomu zajistíme větší bezpečnost při přenášení citlivých dat a můžeme si být jisti, že komunikujeme s ověřenou identitou protistrany. Nevýhodou však je mírné zpomalení oproti protokolu HTTP způsobeno právě kvůli jeho šifrování.

Abychom mohli na našem serveru použít protokol HTTPS, je třeba si zajistit certifikát. Ten může být sdílený, který je většinou zdarma, nicméně nebývá globálně důvěryhodný a prohlížeče budou hlásit problém se zabezpečením, protože nebudou znát použitou certifikační autoritu. Druhou variantou je vlastní certifikát, který vystavují tzv. certifikační autority. České certifikační autority však bohužel nebývají pro internetové prohlížeče důvěryhodné, spíše se využívají pro certifikáty k elektronickým podpisům pro komunikaci se státní správou. (Grill, 2012) Tento certifikát je již plně důvěryhodný a lze ho používat bez jakýkoliv výjimek, nicméně je zpoplatněn a cena se pohybuje řádově ve stovkách až tisících korunách. Tento certifikát navíc není neomezený a po určitě době se musí znovu prodloužit.

2.14.2 ACL

Pro zvýšení bezpečnosti poskytuje Nette Framework vrstvu zvanou Access control list, zkráceně ACL. Díky tomu vzniká určitá hierarchie rolí, které mají určité povolené akce nebo přístupy. Nette obsahuje třídu `Permission`, která poskytuje lehkou a flexibilní ACL vrstvu pro řízení práv a přístupu. (Grudl, 2011) Pomocí této třídy můžeme definovat libovolné uživatelské role, seznam zdrojů, ke kterým mohou uživatelé přistupovat, a pravidla určující, kdo co může s čím dělat. Zdroje i jednotlivé role mohou využívat dědičnosti a není tak nutné psát stále dokola pravidla, které jsou pro všechny role shodné.

3. Analýza a návrh řešení

Analýzu a návrh řešení považuji za jednu z nejdůležitějších částí projektu. Tuto etapu bychom určitě neměli nijak podceňovat, protože od ní se bude vyvíjet všechno ostatní. Protože s obdobným projektem nejsou žádné zkušenosti, je těžké předpovídat časovou analýzu. Nicméně předpokládaná doba vývoje je přibližně šest měsíců. Jelikož se jedná o open-source aplikaci, budou náklady na jeho vývoj vynaloženy pouze za výběr domény pro stránku a za odpovídající hosting. Cena za rok se tak bude pohybovat okolo 500 Kč. Internetový obchod je na zakázku a tak je potřeba nejdříve vykonat analýzu požadavků klienta, která nám určí vlastnosti celého informačního systému. Požadavky můžeme rozdělit na dvě části, funkční požadavky a nefunkční požadavky.

3.1 Funkční požadavky

„*Funkční požadavky představují základní předmět systému a jsou měřeny konkrétními prostředky jako jakou například hodnoty dat, logika a algoritmy rozhodování*“. (Robertson, a další, 1999) Funkční požadavky nám tedy definují, co by měl systém umět a podporovat a jak by měl fungovat. Následná analýza bude použita jako základ takzvané toplevel funkce systému pro funkční analýzu. (DOD, 2001) Pro webovou aplikaci byly stanoveny tyto funkční požadavky:

- Uživatel si bude moct bez potíží vložit produkt do košíku, pokračovat v objednávce a úspěšně ji dokončit. Po dokončení objednávky bude uživateli na jeho email zaslána potvrzovací zpráva s detailním popisem objednávky.
- Uživatel se bude moct registrovat, přihlašovat a odhlašovat. Po registraci se mu odešle na uvedený email potvrzovací zpráva o úspěšné registraci a požádání o dokončení registrace. Aplikace bude moct uživateli poslat nové heslo při zapomenutí starého hesla.
- Aplikace bude moct vyřizovat objednávky i pro nepřihlášené uživatele.
- Aplikace bude obsahovat menu se základními kategoriemi Kojenci, Holky, Kluci, Hry, Kreativita, Stavebnice a Dřevěné hračky. U těchto kategorií bude možné měnit pouze jejich název.
- Po najetí myší na základní kategorii, se zobrazí všechny její podkategorie.

- Aplikace bude poskytovat detailní popis daného produktu, hodnocení, hlavní obrázek, galerii a možnost přidávat komentáře.
- Aplikace umožní uživatelům vyhledávat produkt podle jeho názvu nebo klíčových slov v popisu produktu.
- Veškeré ceny budou uvedeny včetně DPH s možností změnit sazbu.
- Aplikace bude poskytovat neomezené množství produktů a kategorií, které se mohou dále do sebe jednotlivě vnořovat a vytvářet tak určitou hierarchii.
- Přihlášení uživatelé si budou moci měnit své přihlašovací údaje včetně svých kontaktních údajů. Pouze přihlášení uživatelé mohou komentovat a hodnotit produkty.
- Aplikace bude obsahovat jednoduchou administraci, ke které je umožněn přístup autorizovaným uživatelům na základě přidělených práv. V administraci bude moct provozovatel přidávat, měnit a mazat veškeré položky.
- Administrace bude obsahovat tyto prvky, které lze dále upravovat: produkty, kategorie, objednávky, uživatelé, celkové nastavení obchodu (kontakt na prodejce, platba, doprava, slevy) a vzhled.
- Editace administrace nebude požadovat (s výjimkou šablon emailů) znalost žádných programovacích jazyků a bude prováděna jen pomocí internetového prohlížeče.
- Aplikace bude obsahovat bezpečnostní mechanismy.

3.2 Nefunkční požadavky

Nefunkční požadavky nám určují, jaké vlastnosti systém bude mít a definuje určité omezení. Podle Robertsona (1999) tento druh požadavků „*představuje vlastnosti v oblasti chování, které musí mít stanovené funkce, jako například výkonost, uživatelnost atd.* “ Pro webovou aplikaci byly stanoveny tyto nefunkční požadavky:

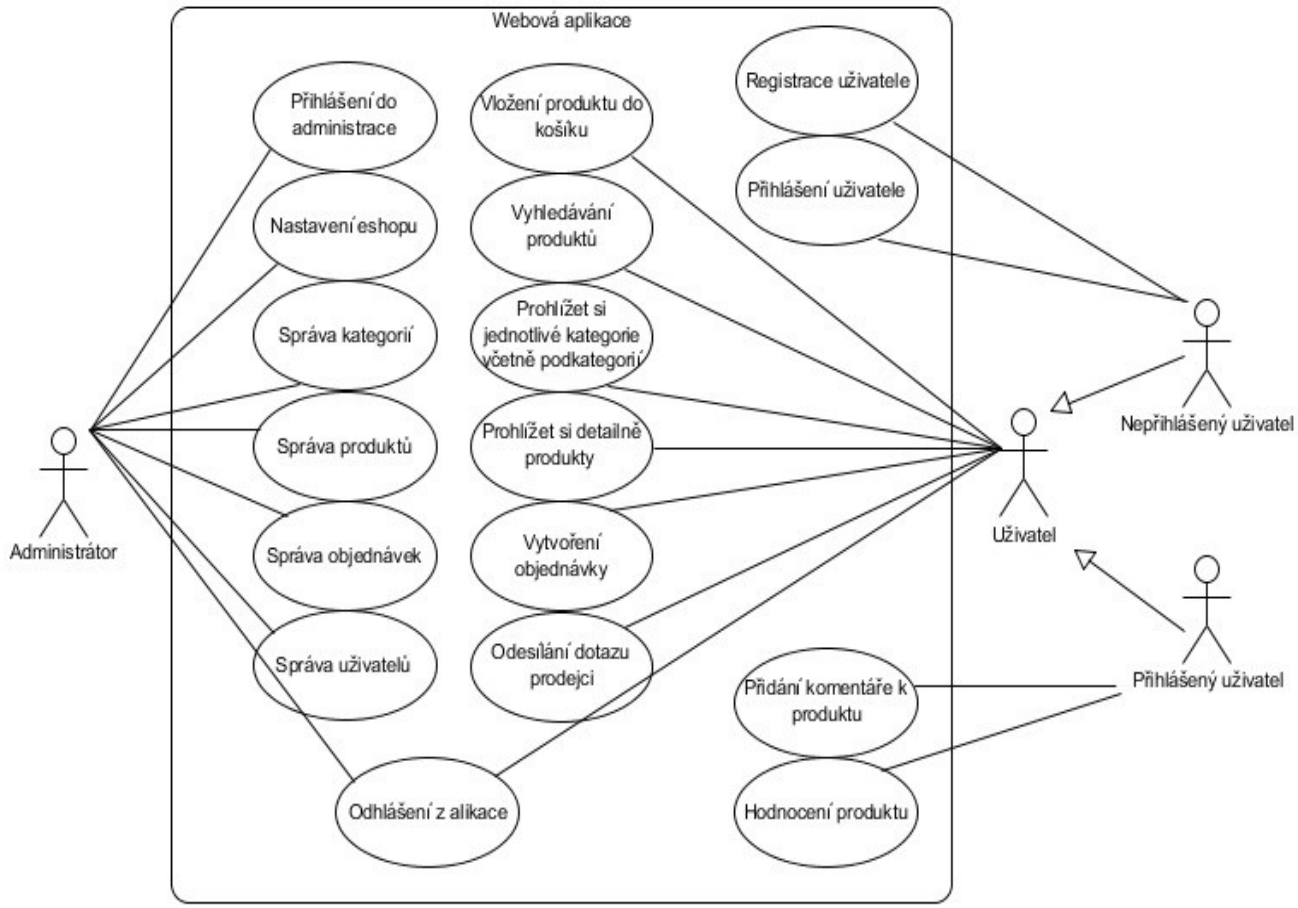
- Aplikaci bude uživatel moct používat ve všech běžně dostupných internetových prohlížečích. Vzhled se bude lišit jen v drobných odchylkách a u starších prohlížečů (např. Internet Explorer 6.0 a nižší) je nejdůležitějších zachování funkcionality.
- Aplikace bude běžet na běžně dostupném veřejném hostitelském serveru podporující všechny systémové požadavky.

- Aplikace bude vytvořena v programovacím jazyce PHP a bude použit Nette Framework.
- Data budou uložena v databázi MySQL.

3.3 Strukturovaná analýza

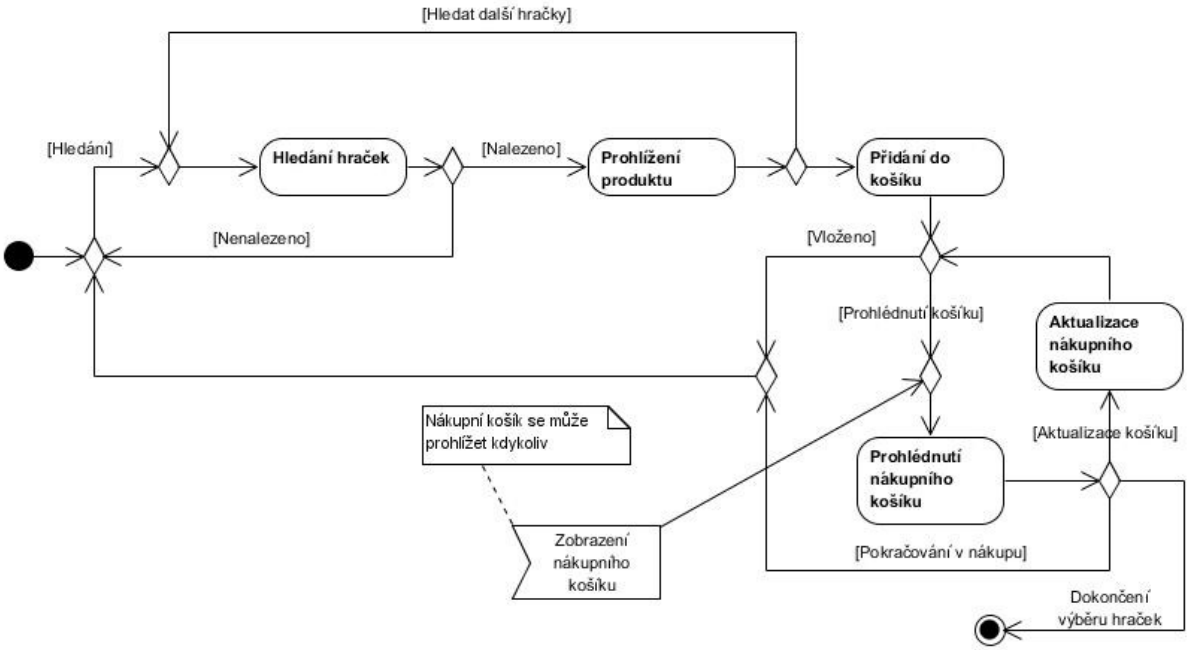
Po analýze funkčních a nefunkčních požadavků je třeba si udělat vlastní visuální návrh řešení, který pomůže jak vývojáři aplikace, tak i klientovi, udělat si přehled o systému a jeho funkcionalitě. K tomuto účelu se využívá unifikovaného modelovacího jazyku UML, který obsahuje sadu grafických notací k tvorbě visuální modelů při vývoji objektově orientovaných systémů.

V našem případě diagram případů užití, zobrazeném na obrázku č. 4, obsahuje aktéry typu administrátor, přihlášený uživatel a nepřihlášený uživatel. Administrátor se přihlašuje do backendové části webové aplikace a má možnost spravovat například nastavení celého eshopu, produkty, kategorie, objednávky, uživatele a tak dále. Dalším účastníkem je přihlášený uživatel, který se přihlásil do frontendové části eshopu a může využívat všech funkcionalit aplikace s výjimkou administrace. Posledním účastníkem je nepřihlášený uživatel, který má možnosti téměř shodné s přihlášeným uživatelem, nicméně si nemůže ukládat zboží do košíku a prohlížet si historii svých objednávek.



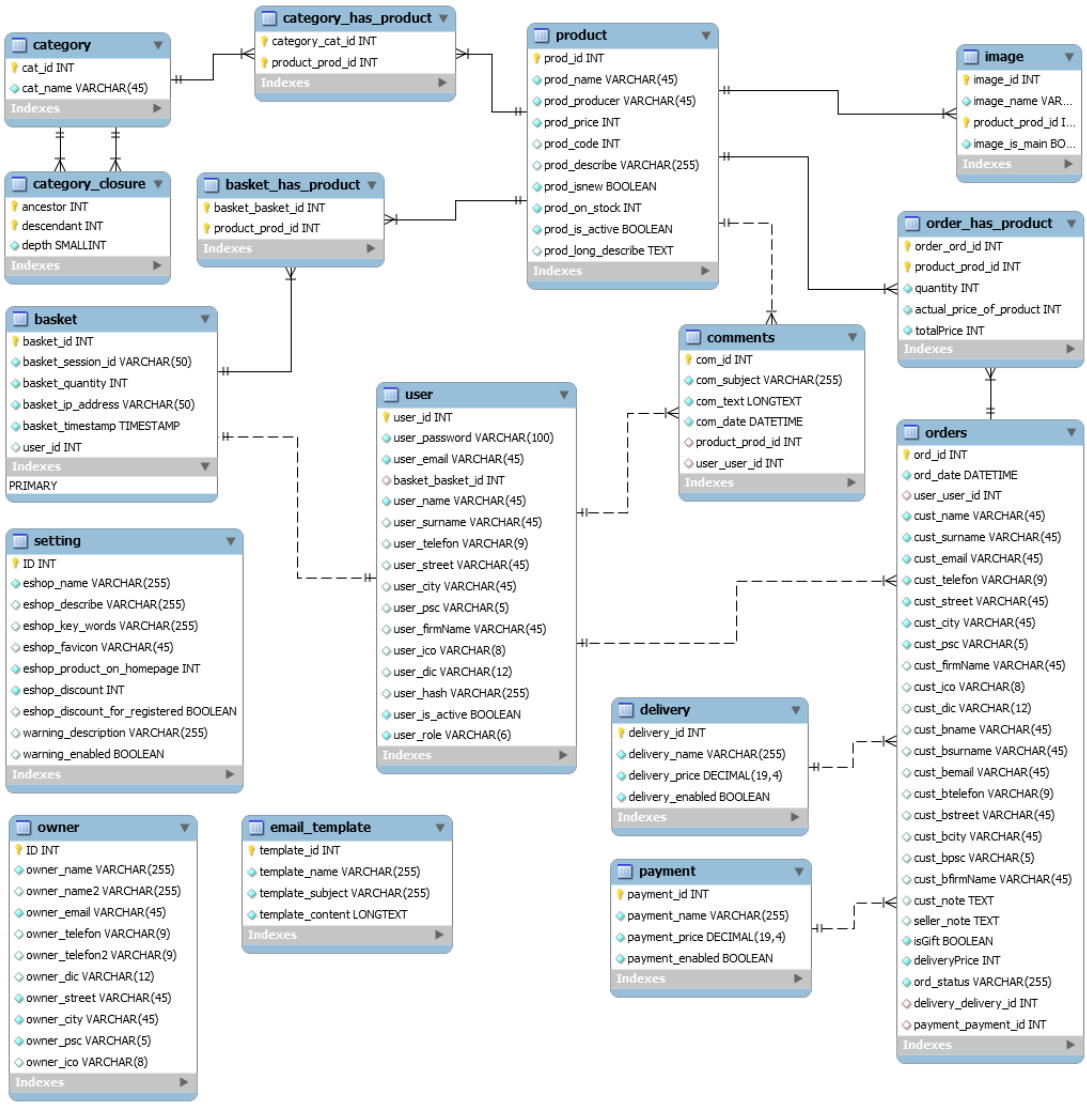
Obrázek č. 4 - Diagram případů užití zahrnující všechny aktéry eshopu

Po analýze aplikace pomocí diagramu případů užití, je potřeba si udělat grafické znázornění pracovních procesů (aktivit) a akcí, které podporují rozhodování, iteraci a souběžnost. Pro tuto analýzu využijeme diagram aktivit, který na rozdíl od diagramu případů užití, dokáže zobrazit paralelní procesy. Například diagram aktivit zobrazený na obrázku č. 5 popisuje proces hledání, prohlížení a vkládání produktu do nákupního košíku. Tento proces je společný jak pro přihlášené tak i pro nepřihlášené uživatele. Uživatel nejprve hledá požadovaný produkt a při jeho nalezení si může prohlídnout detailnější popis. Dále má na výběr, zda si zboží přidá do košíku nebo si najde jiný produkt. Nákupní košík a je automaticky generovaný hned při přístupu uživatele na stránky a je možné si jej prohlédnout kdykoliv. Zde má možnost změnit počet produktů pro budoucí objednávku a aktualizovat tak košík. V případě, že z košíku odstraní všechny produkty, je uživatel upozorněn, že se v něm již žádné zboží nenachází. Uživatel se dále může rozhodnout, zda bude pokračovat v objednávce, či se vrátí zpět k výběru dalších produktů.



Obrázek č. 5 - Diagram aktivit pro výběr produktu a jeho vložení do košíku

Základem všech moderních aplikací, je vytvoření kvalitní databázové základny, ze které bude daná aplikace využívat a spravovat data. Během její tvorby vytváříme konceptuální datový model, který vzniká jako popis dat v databázi bez ohledu na jejich uložení. Díky tomu získáme určitou abstrakci a nezávislost modelu na konkrétní databázi a je tak možné přejít na jiný databázový systém bez nutnosti například celý návrh opakovat. Konceptuální datový model se zobrazuje pomocí grafických vyjádření, konkrétně jako E-R modely nebo jako tzv. diagramy tříd. Podle Kaluži (2011) je „*grafické vyjádření jakéhokoli modelu oproti výrokové formě snáze srozumitelné, přehledné a je přijatelnější i pro uživatele.*“ Jak už z obrázku č. 6 vyplývá, je vytvořeno celkem 16 tabulek, jejich názvy odráží entity skutečného světa, respektive entit, které budeme pro správný chod e-shopu potřebovat.



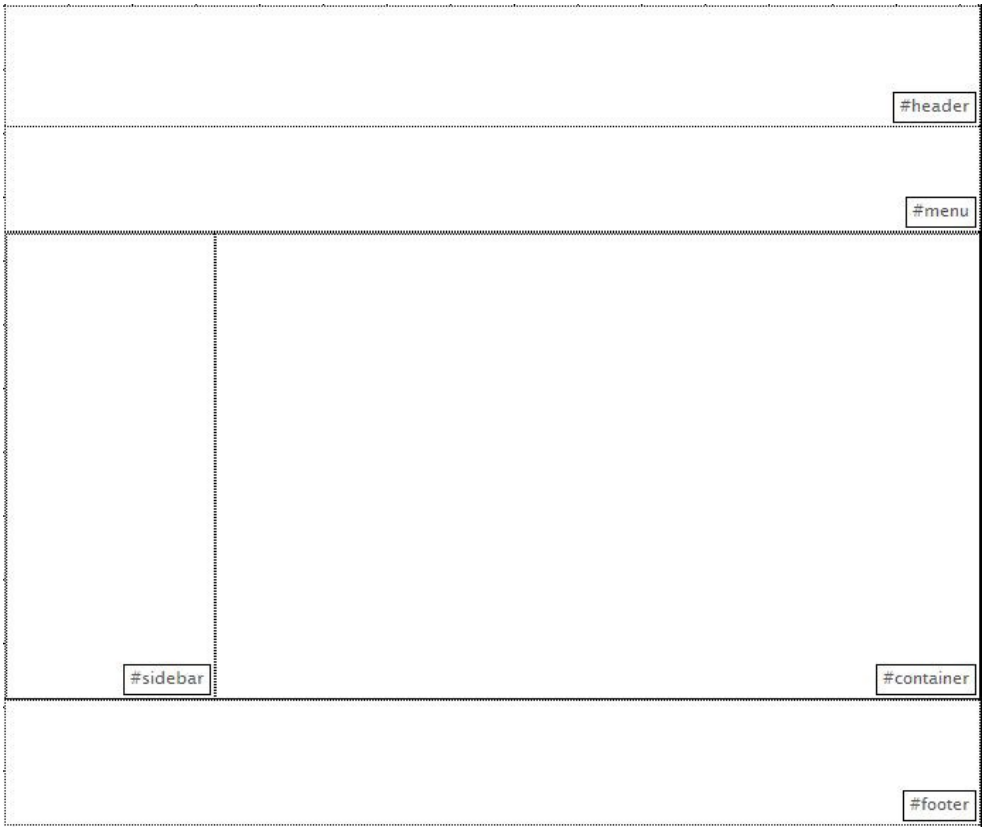
Obrázek č. 6 - E-R diagram internetového obchodu

3.4 Layout

Grafický design internetového obchodu je pro získání nových uživatelů velmi klíčový. Je potřeba zajistit, aby byl vzhled přehledný a zákazník se v něm lehce orientoval. Pro nízkou nákladovost při vytváření internetového obchodu, bude pro front-end využita zdarma dostupná šablona. Ta ovšem obsahuje ale pouze úvodní stránku a bude potřeba dodělat všechny ostatní stránky například pro košík, proces objednávky, detail produktu apod. Dále bude potřeba vyřešit jednoduché prohlížení a vyhledávání a poskytnout možnost vyčerpávajících, podrobných informací o produktech. Pro návštěvníky, kteří využívají starší prohlížeče, především pak Internet Explorer 9 a nižší, bude potřeba plně zpřístupnit web.

Protože front-end design bude vycházet ze šablony, bude zachována i jeho základní struktura, která je zobrazena na obrázku č. 7. Využit bude dvousloupcový layout, kde oba sloupce budou mít pevnou šířku a plovoucí bude pochopitelně pouze jejich výška. Layout bude dále tvořit hlavička, kde se bude nacházet například logo obchodu, vyhledávání nebo přihlašování či registrace. Pod ní se bude nacházet menu, které bude obsahovat všechny kategorie. Ve spodní části layout pak bude patička, ve které se budou nacházet odkazy na podrobnější informace o obchodu, jako například jak mohou zákazníci platit, jaké jsou způsoby doručení, proč se mají zákazníci registrovat apod.

Pro administrační část aplikace bude využit taktéž layout s dvěma sloupci, kde oba budou mít pevnou šířku. V hlavičce bude pouze logo obchodu a možnost odhlásit správce.



Obrázek č. 7 - Návrh front-end layoutu aplikace

4. Realizace a implementace internetového obchodu

Pro realizaci internetového obchodu byl zvolen programovací jazyk PHP především kvůli jeho vysoké rozšířenosti a možnosti ho použít na všech platformách. Protože programování se může časem stát někdy až příliš velkou rutinou, kód se často zbytečně opakuje a některé řešení mohou být velmi komplikované, byla zvolena možnost použití frameworku. Zpočátku padla volba na to, který framework si zvolit, protože jich existuje velké množství a každý se svým způsobem liší. Volba byla nakonec zredukována na dva frameworky – Nette Framework a Zend Framework. Protože ani s jedním nebyla žádná předešlá zkušenost, bylo rozhodnuto raději pro Nette z důvodu českého nástroje a vysoce aktivní komunity, která byla později během vývoje často využívána.

Jako databázový systém bylo zvoleno MySQL hlavně z jeho předešlých zkušeností, svobodného licencování a relativně rychlého výkonu. Nabízely se i jiné proprietární možnosti jako například Oracle nebo Microsoft SQL, nicméně pro jejich velmi nízkou hostingovou podporu a vysoké pořizovací ceně, bylo od této možnosti opuštěno.

4.1 Adresářová struktura

Ve webové aplikaci se vycházelo z adresářové struktury sandboxu, který Nette Framework doporučuje. Během vývoje aplikace se adresářová struktura pomalu rozšiřovala, protože větší komplexnost aplikace vyžadovala i lepší přehled adresářů. Vznikl tak modul AdminModule, který obsahuje veškeré presentery a pohledy co se týče administrace. Přibyl také adresář s názvem components, který obsahuje veškeré komponenty, které aplikace používá. Jedná se o tyto čtyři komponenty, které lze v kterékoliv aplikaci „běžící“ na Nette jednoduše zaregistrovat a používat:

- **Eciovni** – komponenta, díky které můžeme vygenerovat soubor ve formátu PDF a vzhled si upravit v Latte šabloně. Doplněk využívá velmi rozsáhlé knihovny mPDF, kterou jsem vložil do adresáře libs.
- **Rating** – jedná se o komponentu, kterou byla z velké části vytvořena mnou a její hlavní účel je zajistit hodnocení produktů pomocí databáze. Vzhled je možné jednoduše upravit v Latte šabloně, aniž by byla potřeba zasahovat do aplikační logiky komponenty.

- **VisualPaginator** – komponentu naprogramoval samotný autor Nette David Grudl a jedná se o zdokonalení komponenty Paginator, kterou můžeme najít v distribuci Nette Frameworku. Jak už samotný název napovídá, komponenta umožňuje jednoduché stránkování a usnadní nám tak mnoho práce.
- **Bcrypt** – jedná se hashovací algoritmus, který můžeme upravovat podle výkonosti hardwaru. I když PHP samotné poskytuje mnoho zabezpečovacích funkcí, jako jsou například MD5, Blowfish nebo SHA, stále se jedná o kriptovací funkce, které nejsou tolik vhodné pro hashování. Navíc v současnosti existují i tzv. „Rainbow Tables“, které poukazují na nedostatečné zabezpečení cenného hesla jednoduchým hashem a nutí vývojáře k použití mnohem lépe zabezpečených metod. (Jørgensen, 2012) Právě proto vznikl algoritmus Bcrypt, který nutí útočníka použít opravdu masivních prostředků pro prolomení hesla. Bcrypt používá algoritmus Eksblowfish a od „klasického“ Blowfish se odlišuje především tím, že zajišťuje závislost na soli a klíče (v našem případě uživatelské heslo) a prolomení algoritmu pak vyžaduje znalost obojího.

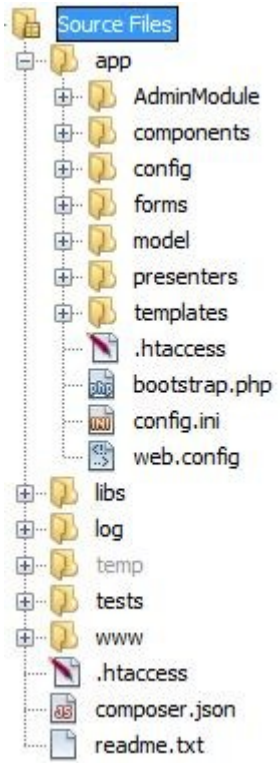
Tím ale celá adresářová hierarchie nekončí. Struktura se skládá celkem ze sedmi základních adresářů. Popíšu ty nejdůležitější pro správnou a nezbytnou funkčnost aplikace.

- Prvním adresářem je `app`, který v sobě zahrnuje celou architekturu MVP, respektive všechny soubory serverové části aplikace. Tudíž jsou v něm veškeré presentery, modely, pohledy, komponenty a moduly aplikace. Adresář navíc obsahuje podadresář config, ve kterém se nalézá hlavní konfigurační soubor `config.neon`, a `bootstrap.php`, zaváděcí soubor celé aplikace.
- „Poháněcím motorem“ celé aplikace je adresář libs. V něm se nachází nejdůležitější část – samotná knihovna Nette. Součástí tohoto adresáře je také soubor `netterobots.txt`, ve kterém máme možnost zvolit si soubory či složky, které nechceme, aby Nette robot indexoval. Také se zde nalézá robustní knihovna mPDF pro generování souborů ve formátu PDF, kterou jsem do adresáře přidal.
- Dalšími, pro vývojáře velmi důležitými, jsou adresáře `temp` a `log`. Do `tempu` můžeme (a Nette to dělá automaticky za nás) přidávat cache či jiné dočasné soubory. Adresář `log` je pak velmi nápomocný v případě, kdy pracujeme v produkčním módu a potřebujeme zjistit, zda nedošlo k nějaké chybě a pokud ano, tak k jaké. Umísťují se zde tedy různé logy, error logy atd. Aby vše fungovalo správně a abychom do adresáře mohli zapisovat, je důležité nastavit práva pro zápis (`chmod 0777`).

- Posledním důležitým adresářem je `www`. Jedná se o veřejný kořenový adresář celé aplikace. Zde se nacházejí všechny CSS soubory, JavaScriptové soubory a obrázky. Jako jediný má být přístupný z vnější strany klienta. Zde se také nalézá klíčový soubor `index.php`, přes který se spouští celá webová aplikace.

Abychom zabránili přístupu z vnější strany klienta, tak se v některých složkách nalézají soubory `.htaccess` pro webový server Apache a `web.config` pro webový server IIS. Tyto soubory zakazují přístup z prohlížeče.

I když se jedná o doporučovanou strukturu adresářů, nejsou zde žádné zábrany pro vytvoření si vlastní adresářové hierarchie.



Obrázek č. 8 - Adresářová struktura aplikace

4.2 Inicializace aplikace

Výchozím bodem aplikace je jediný skriptovací soubor ve složce `www`, a to `index.php`. Protože se soubor nachází až v podadresáři, je třeba v kořenovém adresáři aplikace vytvořit soubor `.htaccess`, který v sobě obsahuje pravidlo pro přesměrování do adresáře `www`:

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteRule ^$ /www/ [L]
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond %{REQUEST_URI} !^/www/
    RewriteRule ^(.*)$ /www/$1
</IfModule>
```

V samotném souboru `index.php` se nacházejí konstanty `WWW_DIR` (absolutní adresa ke kořenovému adresáři), `APP_DIR` (absolutní adresa ke kořenu aplikace, složka `/app`) a `LIBS_DIR` (absolutní cesta ke knihovnám aplikace, složka `/libs`). Na posledním řádku se zde nachází adresa k zaváděcímu souboru aplikace, `bootstrap.php`, kterému je předáno další řízení aplikace.

Prvním úkolem zaváděcího souboru `bootstrap.php` je načíst Nette Framework. Díky tomu můžeme využívat funkce `autoloading`. Následuje načtení třídy `Configurator`, který generuje systémový Dependency Injection (DI) kontejner, který je dále popsán v kapitole 4.2.1. Dále v zaváděcím souboru aktivujeme `Nette Debugger`, neboli „laděnku“, pro vizualizaci chyb a jejich následné logování do souboru `Log`. Poté následuje načtení služby RobotLoader, který si všechny třídy a rozhraní indexuje a ukládá do cache, a v případě dalšího HTTP požadavku jsou tak ihned k dispozici.

Předposlední částí souboru `bootstrap.php` je nastavení routování, které bylo popsáno v kapitole 2.8.4 Routování. Tyto „routy“ budou později nastaveny na konci vývoje aplikace. Poslední částí je pak pouze jediný řádek s `$container->application->run();` který nám spustí aplikaci.

4.2.1 Systémový kontejner

V systémovém kontejneru se nacházejí všechny služby a parametry potřebné pro běh aplikace. Tedy nejen služby frameworku samotného, ale i všech ostatních knihoven, které se rozhodneme použít. Tento DI kontejner se generuje z konfiguračního souboru config.neon, který je ve formátu NEON, sloužící pro serializaci strukturovaných dat do lidsky srozumitelné podoby. V naší aplikaci je zde definována exspirace session na 30 minut a aktivován

debugger bar, který nám pomáhá při vývoji například dumpovat proměnné, sledovat použité databázové dotazy nebo celkový čas načtení stránky.

Asi nejdůležitější částí tohoto souboru je zaregistrování služeb, které budeme v aplikaci potřebovat. Zde jsou zaregistrovány všechny modely, včetně autentizačního modelu a předka `TableModel`, které zajišťuje připojení k databázi. Neméně důležitou částí kontejneru je i definování údajů pro připojení k databázi. Nette umožňuje definovat údaje jak pro produkční mód, tak i pro vývojový mód. Framework automaticky rozpozná, o jaké prostředí se jedná a podle toho se použijí údaje.

4.3 Výsledná grafika

Grafika internetového obchodu vycházela z návrhu, který byl popsán v kapitole 3.4. Jako základ front-endu byla použita zdarma dostupná šablona, která však obsahuje pouze úvodní stranu a bylo potřeba vytvořit vzhled pro ostatní stránky, jako například nákupní košík, proces objednávky nebo detail produktu. Pro základní pozicování stránek byly použity tagy `<div>`, které definují jednotlivé bloky designu. Návrh se mnohokrát měnil z důvodů požadavků klienta a nakonec byla vytvořena základní kostra designu, podle které se vytvořily i ostatní stránky vyjma té úvodní, na které se totiž navíc nachází flash banner prezentující nové produkty a obsahuje tak blok navíc.

Protože front-end návrh vycházel ze šablony, byla ponechána i základní a dominantní barva aplikace, tedy světle zelená, přesněji RGB barva s hexadecimálním číslem `#94a219`. Tato barva produkuje velmi dobrý dojem. S touto barvou souhlasil i klient a tak byla použita pro všechny ostatní stránky, především pak z důvodu jednolitosti barvy celého obchodu. Většinově byla pro text použita černá barva, nicméně pro menu byl pro lepší čitelnost zvolen bílý text. Textový font byl pro celou aplikaci zvolen z rodiny `Arial, Helvetica`.

Pro administrační část aplikace byla použita zdarma dostupná šablona od Muhammada Usmana. Tato šablona vychází z kolekce nástrojů Twitter Bootstrap, který často využívá nejnovějších technologií CSS3 a HTML5. Společně s tímto frameworkem je také spjato používání AJAXu, takže je potřeba, aby správce pro administraci používal moderní internetové prohlížeče, které tyto technologie podporují a využil tak všech možností, které back-end nabízí. Pro administraci byl použit dvousloupcový design, který se opakuje na všech stránkách.

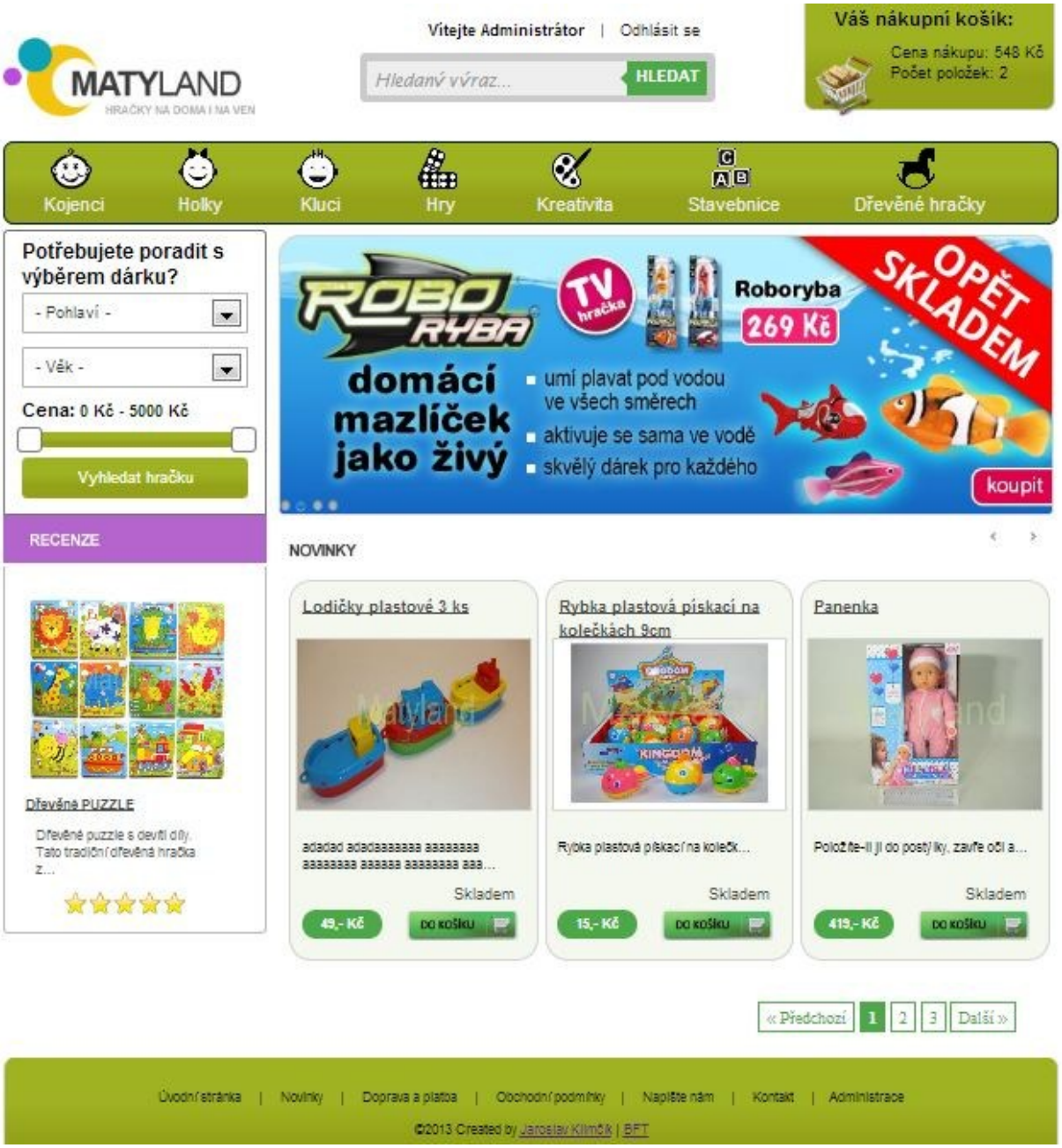
V uživatelském i administračním rozhraní se využívá kaskádových stylů, které se nacházejí v externích souborech pro lepší přehlednost. Pro určité prvky se využívá technologie CSS3, kterou však některé starší prohlížeče nepodporují a je proto potřeba zajistit určitá opatření. Problém se zobrazováním CSS3 má nejčastěji problém prohlížeč Internet Explorer verze 9 a nižší. K vyřešení tohoto problému existují tzv. CSS hacky, které umožňují nastylovat určité bloky pouze pro Internet Explorer. Těchto hacků existuje řada, nicméně asi nejpoužívanější jsou podmíněné komentáře v HTML. V aplikaci je použit tento komentář:

```
<!--[if lte IE 9]>
   <link rel="stylesheet" type="text/css" href="{$basePath}/css/styleIE.css" />
<![endif]-->
```

Zkratka `lte` značí, že se použije externí CSS soubor tehdy, když je použit prohlížeč Internet Explorer verze 9 nebo nižší. V tomto souboru se nacházejí styly, které jsou přímo uzpůsobeny IE prohlížeči a dané prvky jsou pak v něm správně vykresleny.

4.4 Front-end rozhraní

Front-end je ta část webové aplikace, kterou chceme uživateli prezentovat. Klient tak vidí to, co potřebuje a může aplikaci využívat přímo, jako například prohlížení si produktů, nákupního košíku nebo vytvářet objednávky. Podle Jiřího Peterky se front-end „*míní jako něco, co zajišťuje předzpracování či jiné počáteční akce, které lze udělat předem, před vlastním (skutečným, hlavním) zpracováním, co mu může nějakým způsobem odlehčit*“. Front-endová část musí být především přístupná všem uživatelům. Vzhled uživatelské části je zobrazen na obrázku č. 9.



Obrázek č. 9 - Grafika obchodu (front-end)

4.4.1 Zobrazení produktů

Nejnovější produkty na úvodní stránce internetového obchodu má možnost vidět jak přihlášený, tak i nepřihlášený uživatel. Počet zobrazených produktů na úvodní stránce je defaultně nastaven na šest, nicméně lze v administraci zvolit libovolný počet. Od toho se také odvíjí komponenta `VisualPaginator`, který jak už jeho název napovídá, vykresluje a spravuje stránkovač. Tento doplněk usnadnil mnoho starostí s vymýšlením nového algoritmu pro správné a jednoduché stránkování. Krom toho je možné se k produktům dostat pomocí navigačního menu nacházející se na všech stránkách e-shopu. Zde se nacházejí základní kategorie, které byly specifikovány v požadavcích klienta, a po najetí myší na ně, se zobrazují jejich podkategorie. Po jejich zobrazení vidíme všechny produkty, které do dané kategorie spadají.

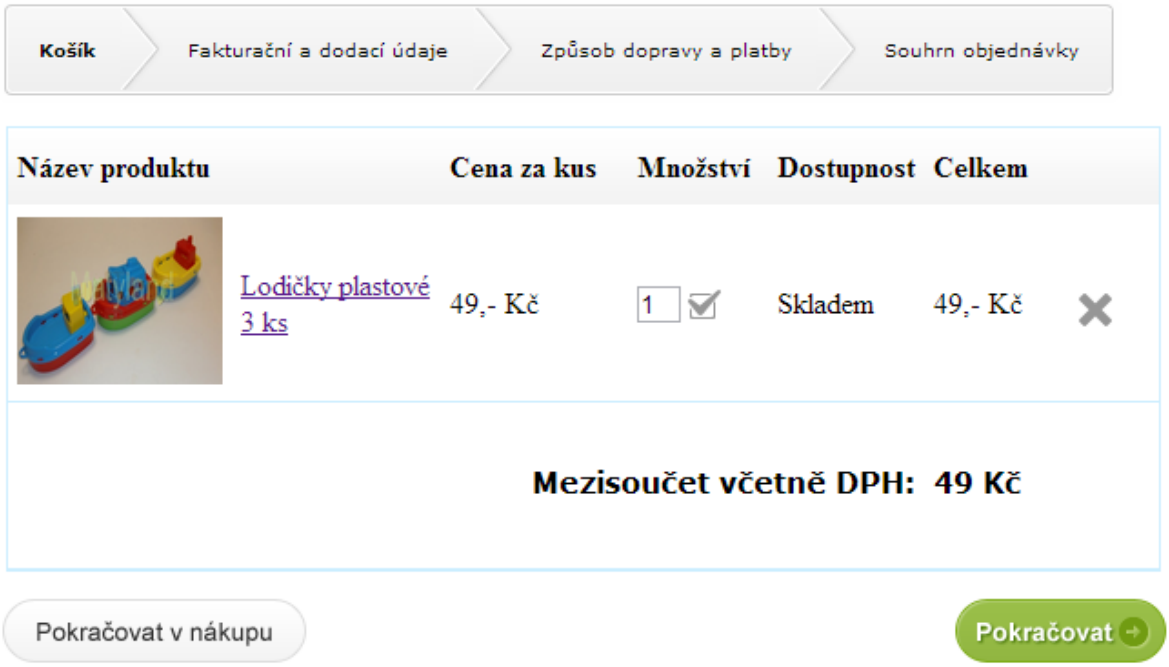
Zjednodušené zobrazení produktu obsahuje jeho název, obrázek a zkrácený popis produktu, který je pomocí helperu `Truncate` zkrácen na 60 znaků. Dále je zobrazena informace, zda je produkt na skladě. V případě, že není, je zobrazen text „Nedostupné“ a není umožněno vkládat produkt do nákupního košíku. V opačném případě lze pomocí tlačítka „Do košíku“ vložit produkt do nákupního košíku a uživatel je následně automaticky do něj přesměrován. Poslední velmi důležitou položkou je cena. Ta je automaticky vypočítána podle výše DPH, které správce obchodu nastavil.

Produkty je možné si také prohlédnou v detailnějším zobrazení, které si můžeme prohlédnout na obrázku, nacházející se v příloze č. 1. Zde se nachází levá navigace obsahující všechny kategorie, nicméně „otevřena“ je pouze ta základní kategorie, do které produkt spadá. V případě, že produkt spadá do určité podkategorie, tak je tato podkategorie viditelně označena. Před samotným detailnějším popisem produktu je drobečková navigace obsahující postupný hierarchický strom kategorií, do nichž produkt spadá. Samotný detail produktu se velmi podobá zjednodušenému zobrazení, ačkoliv přibylo několik prvků. Jedním z nich zobrazení dopravy, respektive za jak dlouho produkt přibližně dojde uživatelovi domů v případě jeho objednání. Text „2-3 pracovní dny“ je statický a nemění se. Druhým novým prvkem je možnost hodnotit produkt. Ten je vykreslen a vykonáván pomocí mé komponenty Rating. Hodnocení je však umožněno pouze registrovaným uživatelům z důvodu lepší ochrany proti útočníkům, kteří by chtěli pouze poškodit majitele e-shopu a úmyslně hodnotit negativně.

Velmi významným prvkem je však tzv. „tabová část“, která obsahuje tři záložky - Podrobnější popis, Obrázky a Komentáře. Za zmínku stojí záložka Komentáře, kde je možné přidávat komentáře pouze přihlášeným uživatelům. Formulář pak obsahuje defaultně jméno uživatele, které nelze měnit. Dále je v poli Nadpis automaticky vložen název produktu, který lze však jakkoliv přepsat. Poslední položkou formuláře je samotný text komentáře.

4.4.2 Nákupní košík

Pro přidání produktu do košíku, je uživatel automaticky přesměrován do nákupního košíku. Ten je automaticky generován podle počtu vložených produktů. V případě, že je vložen jeden produkt vícekrát, zvyšuje se pouze jeho množství v košíku. Jak můžeme vidět na obrázku č. 10, zobrazen je zde název, obrázek, cena za kus, množství, dostupnost a celková cena jednoho produktu. Množství produktu je možné libovolně měnit, nicméně v případě zvolení většího množství, než je na skladě, je do košíku přidán pouze maximálně možné množství a uživatel je na to pomocí flash message upozorněn. V případě, že do množství uživatel nevloží přirozené číslo (kromě nuly), je daný produkt z košíku odstraněn. Vybranou položku lze také z košíku odstranit pomocí křížku na pravé straně tabulky. Současně je i zobrazen mezisoučet cen všech položek včetně DPH.



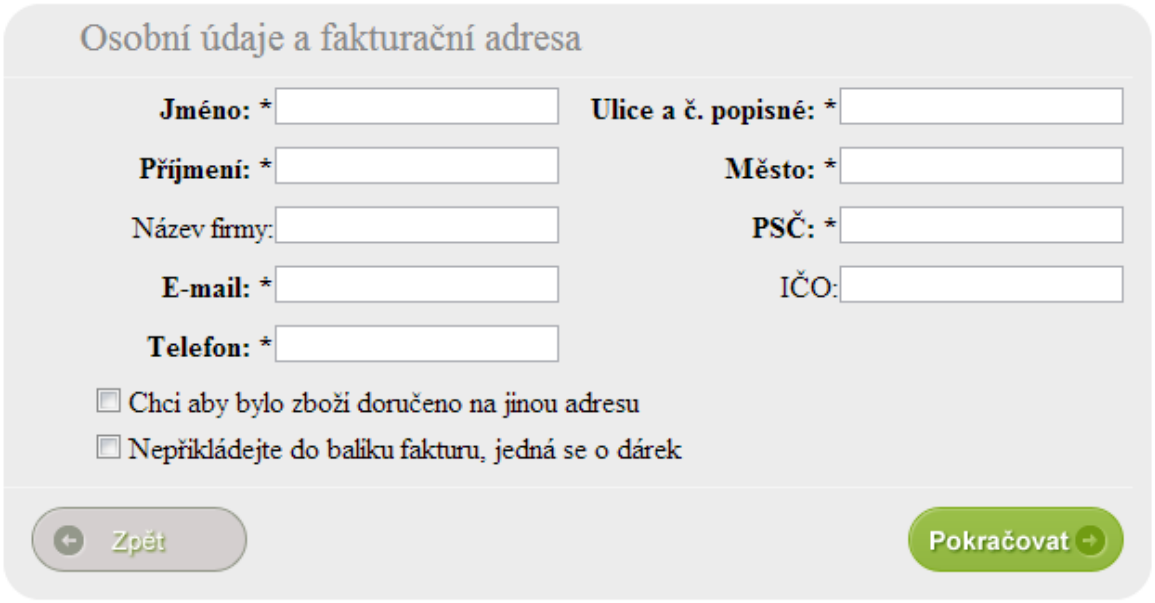
Obrázek č. 10 - Ukázka nákupního košíku

Nákupní košík je založen na session pro jednoduché přenášení všech informací o produktech. V případě nepřihlášeného uživatele má pak obsah košíku v případě neaktivity expiraci 30 minut nebo do zavření internetového prohlížeče. Naopak u přihlášeného uživatele, se veškeré produkty v košíku ukládají do databáze a jsou opět přístupné po jeho opětovném přihlášení. V případě, že měl nepřihlášený uživatel v košíku produkty, a následně se přihlásil, jsou tyto položky automaticky „přeuloženy“ do databáze.

Protože nebyly s tvorbou internetového obchodu žádné předešlé zkušenosti, byla tato část aplikace jedna z nejtěžších. Bylo potřeba promyslet všechny aspekty košíku a možnosti vkládání produktů do něj. Tato implementace zabrala nejvíce času ze všech částí webové aplikace.

4.4.3 Tvorba objednávky

Po vložení produktů do nákupního košíku lze vytvořit objednávku, která má tři etapy - Fakturační a dodací údaje, Způsob dopravy a platby a konečný Souhrn objednávky. V první etapě, zobrazené na obrázku č. 11, je potřeba vyplnit všechny povinné údaje, které jsou označeny hvězdou a tučným písmem. V případě potřeby lze po odškrtnutí checkboxu a vyjetí nabídky doručit na jinou adresu nebo potvrdit, že se jedná o dárek a tudíž správce obchodu nemusí přibalovat do doručovacího balíku fakturu.



Obrázek č. 11 - Ukázka první etapy objednávky

Po vyplnění všech povinný položek, může uživatel přejít na další etapu, tedy do volby dopravy a platby. Tato část je automaticky generována podle volby správce e-shopu. V případě, že klient zvolí špatnou kombinaci dopravy a platby, jako například si zvolí dopravu kurýrem DPD a následně platbu hotově, zobrazí se flash message s upozorněním na špatnou kombinaci a nedovolí mu pokračovat v objednávce.

Posledním krokem je souhrn objednávky. Zde uživatel vidí všechny údaje, které vyplnil, všechny vybrané produkty a celkovou cenu, včetně dopravy a platby. Dále má uživatel možnost napsat prodejci svou poznámku, která se mu následně zobrazí při správě objednávek v administraci. Před samotným dokončením celé objednávky musí uživatel odškrtnout checkbox s odsouhlasením s všeobecnými obchodními podmínkami.

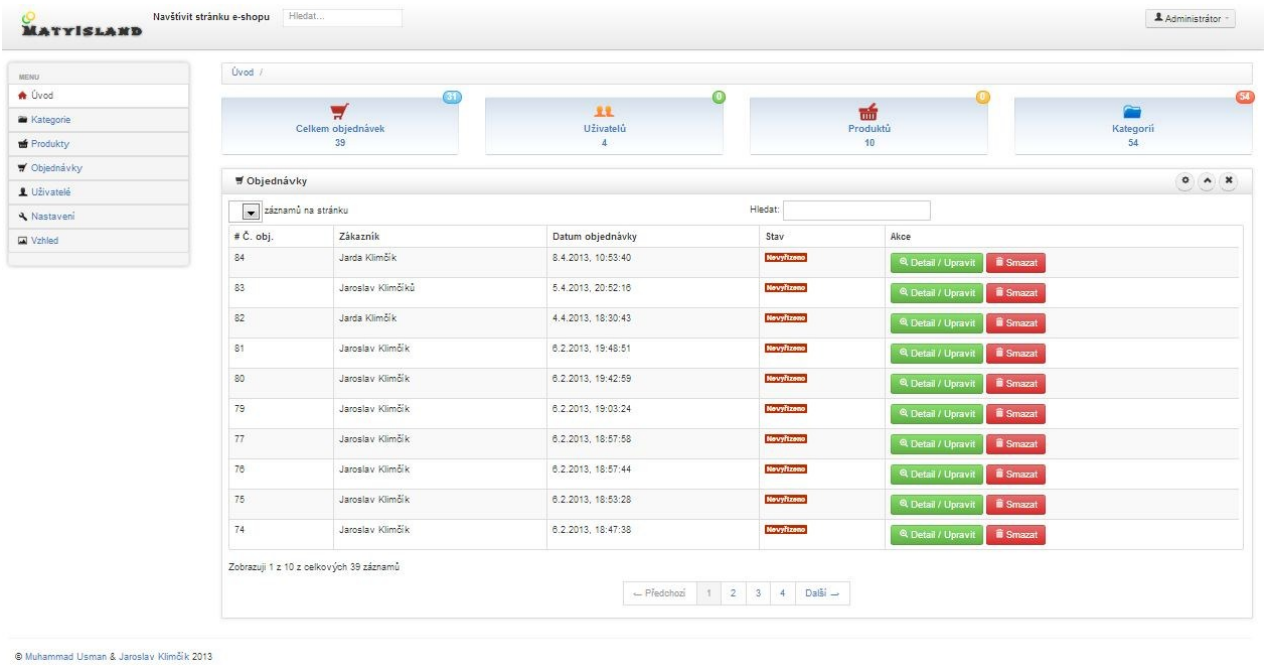
Po kliknutí na tlačítko dokončit, je na uvedený email, který uživatel vyplnil v první etapě objednávky, odeslána zpráva obsahující celkové informace o právě dokončené objednávce. Odesílání emailů je umožněno pomocí třídy Message a není potřeba nijak specifikovat vlastní SMTP server. Následně je uživateli zobrazeno poděkování za nákup a jednoduché zobrazení objednávky včetně jejího čísla a data vytvoření. Nákupní košík se po dokončení objednávky automaticky vyprázdní.

4.4.4 Klientské centrum

Po přihlášení má uživatel možnost navštívit své klientské centrum. Zde nachází základní rozcestník s dvěma možnostmi - Můj účet a Moje objednávky. V sekci „Můj účet“, zobrazeno v příloze č. 2, se nacházejí základní informace o uživateli jako například jméno a příjmení, bydliště, email apod. Navíc se zde nachází možnost změna hesla, kdy je potřeba zadat své stávající heslo, nové heslo a jeho následné potvrzení. Všechny údaje jsou při změně validovány a v případě chyby je uživatel upozorněn pomocí flash message. Druhou částí klientského centra je sekce „Moje objednávky“, kde se nachází historie všech uživatelových objednávek společně s jejich detaily.

4.5 Back-end rozhraní

Back-end je přesným opakem front-endu. Jedná se o tu část aplikace, kde se nachází administrační rozhraní e-shopu. Běžnému návštěvníkovi je tato část skrytá a přístup k ní má pouze správce internetového obchodu. Vzhled back-endu je zobrazen na obrázku č. 13.



Obrázek č. 12 - Vzhled administrační části e-shopu

Aby byla aplikace přehledná, bylo administrační rozhraní vytvořeno v samostatném modulu, který se v adresářové struktuře nachází v `app/AdminModule`. Zde se nacházejí všechny presentery, pohledy a formuláře, které s tímto modulem souvisejí. Modelovou část jsem nechal společnou pro front-end i pro back-end z důvodů aby zbytečně nevznikala duplicita modelových entit a následně případné kolize v případě nepřidání správných jmenných prostorů.

Do back-end rozhraní se správce dostane pomocí odkazu nacházející se v patičce obchodu. Tento odkaz je zobrazen pouze správci po jeho přihlášení. Žádný jiný přihlášený nebo nepřihlášený uživatel ho nevidí. V případě, že by se pokusil útočník navštívit administrační rozhraní i bez použití odkazu určeného pouze pro správce, tedy zadáním správné URL adresy, je tento útočník automaticky přesměrován na úvodní stránku e-shopu, aniž by o tom byl jakkoliv informován. O rozlišení mezi uživatelem aplikace a správcem se stará systém ACL, kterým Nette Framework disponuje.

Na úvodní stránce administrace se nachází navigační menu s jednotlivými kategoriemi pro obsluhu obchodu. Zde jsou uvedeny nejnovější objednávky a jejich krátký popis a celkový počet nevyřízených objednávek, zablokovaných nebo neaktivovaných uživatelů, neaktivních produktů a celkový počet všech kategorií.

4.5.1 Správa kategorií

V tabulkovém výpisu kategorií se nacházejí jen ty základní. Ty nelze smazat, pouze přejmenovat a lze do nich vkládat další podkategorie. V tabulce jsou také uvedeny doplňující informace o počtu produktů a podkategorií obsažených v dané kategorii. Podkategorie jsou vypisovány stejně jako ty základní kategorie, navíc však přibyla možnost je smazat. Smazání je ještě jednou potřeba potvrdit v modálním okně, obsahující varování, že v případě, že kategorie obsahuje produkty, které jsou pouze v jedné kategorii, budou tyto produkty bez zařazení a je potřeba je začlenit do jiné kategorie.

Úroveň zanořování kategorií je libovolná, nicméně pro lepší přehlednost je doporučováno vytvářet do třetí úrovně. Možnost vícenásobného zanořování je umožněna pomocí rekurzivního volání stejného pohledu.

4.5.2 Správa produktů

Produkty jsou zobrazeny v tabulce a seřazeny od nejnovějších, resp. podle jejich ID. V tabulce se nacházejí mimo jiné informace o jejich počtu na skladě, a zda jsou zveřejněny či ne. V případě nezveřejnění jsou tyto produkty pro návštěvníky e-shopu neviditelné a není k nim přístup. Zobrazeny jsou pouze v historii uživatelových objednávek.

V samotném detailu produktu může správce měnit jeho název, množství na skladě, zda se jedná o novinku nebo zda má být zveřejněn. Dále je možné přidávat produkt do více kategorií. Upravena může být i cena včetně informativního zobrazení ceny bez DPH. V další části správy následuje krátký a dlouhý popis produktu. To je umožněno pomocí komponenty TinyMCE a administrátor si tak může zvolit vlastní formátování a vzhled textu, který je pak návštěvníkovi zobrazen. Komponenta je navíce kompletně v češtině a je velmi intuitivní. Poslední částí jsou obrázky produktu. Těch lze k produktu přidávat libovolně, nicméně hlavní obrázek může existovat pouze jeden. Jedná se o ten obrázek, který je zobrazen u produktu jako úvodní. Při najetí myší na obrázek se objeví panel s dvěma tlačítky, jedním pro nastavení obrázku jako hlavního, a s druhým pro jeho smazání.

4.5.3 Správa objednávek

Objednávky jsou zobrazeny v tabulce, která je shodná s tabulkou nacházející se na úvodní stránce administrace. Obsahuje informace o čísle objednávky, jménu a příjmení zákazníka, datu a čase objednávky a jaký je aktuální stav objednávky.

Detail objednávky obsahuje fakturační a dodací údaje zákazníka. Ty je možné v modálním okně změnit. Další důležitou částí je zobrazení položek v objednávce. Zde jsou uvedeny všechny vybrané produkty, způsob dopravy, placení a jejich ceny včetně pak celkové ceny. Položkám v objednávce je možné upravit jejich množství nebo je zcela z objednávky odebrat. Lze i naopak další položku přidat pomocí modálního okna, kde se nachází pole pro vyhledávání produktu. Po vybrání produktu se i automaticky vybere jeho cena. To je umožněno pomocí pluginu do Twitter Bootstrap s názvem Typeaheadmap. Dále je možné opět v modálním okně změnit způsob doručení a placení.

V pravé části detailu objednávky se nachází informace o jejím datu a času vytvoření, zda se jedná o dárek (a tudíž není potřeba přibalovat fakturu) a zda uživatel napsal majiteli obchodu nějakou zprávu. Další částí je aktuální stav objednávky. Stav má informativní charakter pro správce, který má možnost zvolit si jednu z pěti možností stavu - vyřízeno, nevyřízeno, expedováno, stornováno a vráceno. Poslední částí detailu je možnost nechat si automaticky vygenerovat fakturu objednávky nebo napsat zákazníkovi email. Faktura se generuje do PDF formátu pomocí komponenty Eciovni.

4.5.4 Správa uživatelů

Tak jako všechny předchozí správy, jsou i uživatele vypisováni v tabulce obsahující informace o ID uživatele, jeho jménu a příjmení, emailu a statusu, který má tři různé stavy. Po registraci uživatele je na jeho email zaslán potvrzovací email, čímž kompletně dokončí registraci. Do té doby je však jeho status „Neaktivní“ a nemůže se přihlásit do aplikace. V případě dokončení se jeho stav automaticky mění na „Aktivní“. V případě, že uživatele správce zablokuje, získá status „Zablokován“ a není mu umožněno přihlášení.

V detailu uživatelské správy lze měnit uživatelovy údaje, včetně jeho přihlašovacího emailu. Heslo samozřejmě správci zobrazeno není a nemá k němu ani žádný přístup. Ve spodní části se pak nachází historie uživatelem vytvořených objednávek včetně možnosti „přesunout se“ přímo na její detail do sekce Správy objednávek.

4.5.5 Nastavení a vzhled

V sekci Nastavení má správce na výběr z pěti částí. První z nich je „Nastavení e-shopu“, který obsahuje název e-shopu (zobrazeno v tagu `title`), jeho popis (meta tag `description`) a klíčová slova (meta tag `keywords`), sloužící spíše pro vylepšení SEO. Dále

je zde možnost měnit favicon a počet produktů na hlavní stránce. Druhou částí je „Prodejce“, která obsahuje veškeré informace o prodejci. Podle informací o bydlišti je pak automaticky vytvořen náhled v Google Maps, který se nachází ve front-endu v sekci Kontakty. Třetí a čtvrtou částí je pak „Platba“ a „Doprava“. Zde má správce možnost nastavit popis k danému způsobu dopravy či platby, jeho příplatek nebo případně celou službu deaktivovat. Poslední částí je pak „Sleva“. Zde e-shop umožňuje zvolit si procentní slevu na všechny produkty. Ta se návštěvníkovi obchodu zobrazí jako nová cena, zatímco ta stará cena bude přeškrtnuta. Nula pak znamená žádnou slevu.

Sekce Vzhled obsahuje pouze dvě částí, a to „Oznámení na hlavní stránce“ a „Šablony emailů“. V první části je si možné pomocí komponenty TinyMCE vytvořit vlastní text upozornění a v checkboxu si zvolit, zda zapnout či vypnout upozornění. Ve druhé části jsou vypsány všechny šablony emailů, které jsou odesílány uživatelům například při vytvoření nového účtu, resetování hesla nebo při potvrzení dokončené objednávky. Tyto šablony má pak správce možnost editovat.

4.6 Použité routování

Jak bylo popsáno v teoretické části, tzv. „routy“, ovlivňující vzhled URL adres, se nastavují v souboru bootstrap.php. Jako první je nastavena maska pro název produktu, která nám zajišťuje, že v adrese nebude žádná diakritika.

```
Route::addStyle('titleProduct');
Route::setStyleProperty('titleProduct', Route::FILTER_OUT, function($url) {
            return Strings::webalize($url);
        });
```

Tyto masky je možné dědit a toho bylo také využito při nastavení masky pro název kategorie, kde je rovněž diakritika nežádoucí.

Následuje ověření, zda internetový obchod „běží“ na webovém serveru Apache a zda-li podporuje modul FastCGI, který je pro webhosting hojně rozšířen. Pokud není v aplikaci definovaná žádná routovací maska nebo server nepodporuje mod_rewrite, je použit výchozí `SimpleRouter`. Ten dokáže také generovat i přijímat URL adresy, nicméně nedokáže vytvářet „user-friendly URL“ protože nemá jako jeden z parametrů masku tvaru vstupní/výstupní URL. Ty pak mají tvar klasického query stringu. Výsledná adresa pak může vypadat například takto:

```
http://toy.cz/index.php?presenter=product&action=default&id=11&titleProduct=hracka
```

Pokud bychom si však chtěli nadefinovat vlastní cestu a náš server podporuje povolení mod_write, je třeba vytvořit masku. Jako příklad můžeme použít předešlou ukázku URL adresy a chceme, aby výsledek vypadal následovně:

```
http://toys.cz/product/11-hracka/
```

Z tohoto URL požadavku vidíme, že první entita, kterou požadujeme, se nachází za prvním lomítkem. Jak z názvu vyplývá, požadujeme řadič, respektive presenter, s názvem Product. Protože za další lomenou čárou nenachází žádná akce, považuje se za výchozí akce s názvem Default, která se pak v adrese explicitně nezobrazuje. Další a poslední částí jsou parametry, které se akci předávají. V našem případě se jedná o parametr ID s hodnotou 11 a parametr titleProduct s hodnotou „hracka“, který odpovídá skutečnému názvu produktu. Všechny routy se definují v souboru bootstrap.php, který spouští celou webovou aplikaci. Abychom dosáhli kýženého výsledku, bylo potřeba vytvořit jednoduchou masku, která bude splňovat naše požadavky na tvar adresy:

```
router[] = new Route('product/<id>-<titleProduct>', array(
                     'presenter' => 'Product',
                     'action' => 'default'));
```

Protože byly zdrojové kódy psány v angličtině, bylo třeba přeložit své routy do češtiny. Důvodem byl také jednoduchý přechod na anglickou verzi obchodu v případě přidání další jazykové mutace. K jeho dosažení byl využit překladový slovník Nette Frameworku a definice v poli rozšířena:

```
$route = new Route('<presenter>/<action>/<id>', array(
    'presenter' => array(
        Route::VALUE => 'Homepage',
        Route::FILTER_TABLE => array(
            'produkt' => 'Product',
            'kosik' => 'Cart',
            'kategorie' => 'Category'),
    ),
    'action' => 'default',
    'id' => NULL,
));
```

Překladovou tabulku lze tímto způsobem aplikovat na jakýkoliv parametr. Přičemž nefunguje jako filtr, tj. pokud překlad existuje, přeloží se, a pokud neexistuje, bere se původní hodnota.

4.7 Zabezpečení aplikace

Při přihlašování a registraci je důležité dbát na správné zabezpečení hesla, díky kterému se uživatel do aplikace přihlašuje. To je zajištěno pomocí skriptu Bscript, který byl detailněji popsán v kapitole 4.1 Adresářová struktura.

Po přihlášení uživatele je automaticky podle databáze rozpoznáno, jakou roli má uživatel. To je zajištěno pomocí technologie ACL, které definuje skupiny rolí pro uživatele, kteří mají práva užívat určité nadefinované zdroje. V aplikaci jsou definovány pouze dvě role a to „Admin“ a „Member“. Jak už z jejich názvů vyplývá, role „Member“ značí přihlášeného uživatele, který má možnost editovat si vlastní účet, ukládat si položky do košíku, hodnotit produkty apod. Role „Admin“ dědí stejné pravidla jako role „Member“, nicméně má navíc možnost přístupu do administrace. Tato role je uložena v relační databázi ve sloupci `user_role` a po přihlášení uživatele je k němu automaticky přidána.

Celá administrace je pak zabezpečena pomocí šifrovacího protokolu HTTPS a není tak potřeba bát se cizího „odposlechu“ dat. Tento zabezpečovací protokol používám v administrační části aplikace se sdíleným certifikátem z důvodu co nejnižších nákladů. Správce má pak při hlášení prohlížeče kvůli nedůvěryhodnosti certifikátu dvě možnosti - buď bude chybu ignorovat a v prohlížeči schválí výjimku nebo si do svého počítače nainstaluje kořenový certifikát internetového hostingu.

5. Závěr

Tato práce si nečiní za cíl vytvořit „ten nejlepší“ internetový obchod, kterých je na trhu nespočet. Snaží se spíše postavit solidní základ pro internetové obchodování s důrazem na jednoduchost, použitelnost a bezpečnost. To vše s přihlédnutím k důkladné analýze principů elektronické komerce, bezpečnostních rizik a použitím moderních technologií a postupů, jako je návrhový vzor MVP (MVC) nebo ORM vrstva pro práci s objekty nad relační databází.

V úvodní kapitole byl popsán elektronický obchod, který dnes představuje velmi populární způsob podnikání. Byl uveden důvod tvorby vlastního e-shopu a výběr frameworku.

Ve druhé kapitole této bakalářské práce byly popsány teoretické východiska aplikace, které popisují technologie použité v rámci tvorby internetového obchodu. Společně s nimi byly popsány všechny využívané knihovny a další prostředky, které byly potřebné pro vývoj aplikace.

Kapitola třetí popisuje funkční a nefunkční požadavky klienta, které definovaly, co by měl systém umět, podporovat, jak by měl fungovat a jaké vlastnosti a omezení bude. Pomocí strukturované analýzy bylo potřeba si udělat vlastní visuální návrh řešení, který byl popsán pomocí diagramu případů užití a diagramu aktivit. Návrh databázového modelu byl graficky vyjádřen pomocí E-R modelu. V této kapitole byla navržena základní kostra vzhledu obchodu.

Samotná realizace a implementace internetového obchodu byla popsána ve čtvrté kapitole. Zde byly popsány základní postupy pro tvorbu obchodu s využitím frameworku Nette. Byla popsána upravena adresářová struktura aplikace a její inicializace, včetně systémového kontejneru, který obsahuje všechny potřebné služby pro správný chod e-shopu. Byl zobrazen a popsán výsledný vzhled uživatelské a administrátorské části. Kapitola se detailně zabývá jednotlivým částem e-shopu, od zobrazení produktů, nákupního košíku, tvorby objednávky a klientského centra až po administraci a její správu. Dále pojednává o použitém routování URL adres a bezpečnosti e-shopu.

Cílem bylo zrealizovat funkční elektronický obchod, který bude odpovídat požadavkům klienta a zákazníkovi nabídne jednoduché a rychlé nakupování, včetně přehledné orientace v obchodě. Zákazník tak může procházet jednotlivé kategorie včetně jejich podkategorií, vyhledávat zboží a zobrazovat detailní informace o nich. Produkty může vkládat do košíku a

následně vytvořit objednávku. Na druhou stranu administrace poskytuje správci velkou kontrolu nad správou obchodu. Může vytvářet, upravovat a mazat kategorie, produkty, objednávky a uživatele. Dále může nastavovat způsob placení a dopravy, měnit údaje o prodejci nebo měnit vzhled emailových šablon. Díky těmto možnostem je zcela naplněn cíl, jenž byl na začátku práce stanoven.

Internetový obchod byl vyvíjen pomocí Nette Frameworku, který velmi usnadnil jeho tvorbu. Nette vychází z architektury Model-View-Presenter, který izoluje business logiku od uživatelského rozhraní, umožňující upravovat jednotlivé části odděleně. Díky tomu je možné aplikaci relativně jednoduše rozšířit a nabídnout nové funkce, případně kompletně změnit vzhled e-shopu.

I když je systém plně funkční, nebyl e-shop při dokončení této práce stále nasazen do produkčního prostředí a vystaven tak většímu zatížení klientských požadavků, takže je možné, že bude třeba ještě některé věci upravit. Do budoucna by bylo vhodné zajistit také lepší SEO optimalizaci, propojení se zbožovými vyhledávači nebo dokonalejší vyhledávání v e-shopu.

Seznam použité literatury

Bernard, Borek. 2009. Prezentační vzory z rodiny MVC. *Zdrojak.cz.* [Online] 11. Květen 2009. [Citace: 10. Březen 2013.] http://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/.

Berners-Lee, Tim. 1998. Cool URIs don't change. [Online] W3C, 1998. [Citace: 7. Duben 2013.] http://www.w3.org/Provider/Style/URI.html.

Berners-Lee, Tim. 1989. Information Management: A Proposal. [Online] CERN, Březen 1989. http://www.w3.org/History/1989/proposal.html.

Böhmer, Marian. 2012. *Návrhové vzory v PHP.* Brno : Computer Press, 2012. ISBN 978-80-251-3338-5.

Bos, Bert. 2011. CSS current work. [Online] World Wide Web Consortium, 26. Únor 2011. http://www.w3.org/Style/CSS/current-work.

Bos, Bert. 2011. Descriptions of all CSS specifications. [Online] World Wide Web Consortium, 18. Únor 2011. http://www.w3.org/Style/CSS/specs.

Darie, Cristian a Brinzarea, Bogdan. 2006. *AJAX a PHP: tvoříme interaktivní webové aplikace profesionálně.* Brno : Zoner Press, 2006. ISBN 80-868-1547-1.

DOD. 2001. Systems Engineering Fundamentals. *Defense Acquisition University.* [Online] Leden 2001. [Citace: 10. Březen 2013.] http://www.dau.mil/pubs/pdf/SEFGuide%2001-01.pdf.

Gilmore, Jason. 2007. *Velká kniha PHP a MySQL 5: kompendium znalostí pro začátečníky i profesionály.* Brno : Zoner Press, 2007. ISBN 80-868-1553-6.

Goldman, David. 2012. HTML5: The future of the Web is finally here. [Online] CNNMoney, 17. 12 2012. http://money.cnn.com/2012/12/17/technology/html5/index.html.

Grill, Zdeňka. 2012. Základy HTTPS (HTTP přes SSL). *wedos.cz.* [Online] 30. Březen 2012. [Citace: 7. Duben 2013.] http://kb.wedos.com/webhosting/https-zaklady.html.

Grudl, David. 2009. Kdo používá Nette Framework? *Nette.org.* [Online] 13. Březen 2009. [Citace: 10. Březen 2013.] http://nette.org/cs/kdo-pouziva-nette-framework.

Grudl, David. 2011. Konfigurace. *nette.org.* [Online] 2. Červenec 2011. [Citace: 23. Březen 2013.] http://doc.nette.org/cs/configuring.

Grudl, David. 2011. NEON. *ne-on.org.* [Online] Nette Foundation, 2011. [Citace: 23. Březen 2013.] http://ne-on.org/.

Grudl, David. 2009. Nette Framework: adresářová struktura aplikace. *zdrojak.cz.* [Online] 14. Duben 2009. [Citace: 23. Březen 2013.] http://www.zdrojak.cz/clanky/nette-framework-adresarova-struktura-aplikace/.

Grudl, David. 2009. Nette Framework: zvyšte svoji produktivitu. *Zdrojak.cz.* [Online] 10. Březen 2009. [Citace: 10. Březen 2013.] http://www.zdrojak.cz/clanky/nette-framework-zvyste-svoji-produktivitu/.

Grudl, David. 2011. Přihlašování & oprávnění uživatelů. *nette.org.* [Online] 24. Červen 2011. [Citace: 7. Duben 2013.] http://doc.nette.org/cs/security.

Grudl, David. 2011. Routing. *nette.org.* [Online] 18. Červen 2011. [Citace: 7. Duben 2013.] http://doc.nette.org/cs/routing.

Grudl, David. 2011. Šablony. *Nette.org.* [Online] 14. Červen 2011. [Citace: 10. Březen 2013.] http://doc.nette.org/cs/templating.

Grudl, David. 2011. Výchozí helpery šablon. *nette.org.* [Online] 14. Červen 2011. [Citace: 25. Březen 2013.] http://doc.nette.org/cs/default-helpers.

Jørgensen, Martin. 2012. Introduction to Rainbow Tables. *freerainbowtables.com.* [Online] 19. Říjen 2012. [Citace: 21. Březen 2013.] http://www.freerainbowtables.com/articles/introduction_to_rainbow_tables/.

Kaluža, Jindřich a Kalužová, Ludmila. 2011. *Modelování dat v informačních systémech.* Praha : Ekopress, 2011. ISBN 978-80-86929-81-1.

Karwin, Bill. 2010. Models for hierarchical data. *slideshare.net.* [Online] 20. Květen 2010. [Citace: 6. Duben 2013.] http://www.slideshare.net/billkarwin/models-for-hierarchical-data.

Lecky-Thompson, Ed a Nowicki, Steven. 2010. *PHP 6: programujeme profesionálně.* Brno : Computer Press, 2010. ISBN 978-80-251-3127-5.

Peterka, Jiří. 1994. Front end. *ComputerWorld.* 1994, 8.

Procházka, Filip. 2011. Databáze a model. *nette.org.* [Online] 23. Prosinec 2011. [Citace: 24. Březen 2013.] http://doc.nette.org/cs/quickstart/database.

Robertson, Suzanne a Robertson, James. 1999. *Mastering the Requirements Process.* Londýn : Addison-Wesley, 1999. ISBN 0-201-36046-2.

Schlossnagle, George. 2004. *Pokročilé programování v PHP 5.* Brno : Zoner Press, 2004. ISBN 80-868-1514-5.

Van Duyne, Douglas, Landay, James a Hong, Jason. 2005. *Návrh a tvorba webů: vytváříme zákaznicky orientovaný web.* Brno : CP Books, 2005. ISBN 80-251-0508-3.

van Kesteren, Anne a Pieters, Simon. 2011. HTML5 differences from HTML4. *w3.org.* [Online] 5. Duben 2011. [Citace: 20. Březen 2013.] http://www.w3.org/TR/2011/WD-html5-diff-20110405/.

Vávra, David. 2012. GitHub for Windows: brána do světa Gitu bez překážek. *zdrojak.cz.* [Online] 4. Červenec 2012. [Citace: 7. Duben 2013.] http://www.zdrojak.cz/clanky/github-for-windows-brana-do-sveta-gitu-bez-prekazek/.

Voráček, Jan. 2012. Closure Table - stromy v MySQL trochu jinak. *blog.voracek.net.* [Online] 29. Leden 2012. [Citace: 6. Duben 2013.] http://blog.voracek.net/databaze/closure-table-stromy-v-mysql-trochu-jinak/.

Vrána, Jakub. 2011. NotORM v Nette. *PHP triky.* [Online] 31. Leden 2011. [Citace: 10. Březen 2013.] http://php.vrana.cz/notorm-v-nette.php.

Zelenka, Petr. 2005. Metody ukládání stromových dat v relačních databázích. *interval.cz.* [Online] 2. Březen 2005. [Citace: 6. Duben 2013.] http://interval.cz/clanky/metody-ukladani-stromovych-dat-v-relacnich-databazich/.

Seznam zkratek

ACL	Access Control List
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASP	Active Server Pages
CERN	evropská organizace pro jaderný výzkum
CMS	Content Management System
CSRF	Cross-site Request Forgery
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
JSP	JavaServer Pages
MVC	architektura Model–View–Controller
MVP	architektura Model–View–Presenter
ORM	Object Relational Mapping
PEAR	PHP Extension and Application Repository
PHP	PHP: Hypertext Preprocessor (dříve Personal Home Page)
RDBMS	Relational Database Management System
SEO	Search Engine Optimization
SQL	Structured Query Language
SSL	Secure Sockets Layer
SVG	Scalable Vector Graphics

TLS	Transport Layer Security
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WHATWG	Web Hypertext Application Technology Working Group
WYSIWYG	akronym z anglického „What you see is what you get“
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations
XSS	Cross Site Scripting

Prohlášení o využití výsledků bakalářské práce

Prohlašuji, že

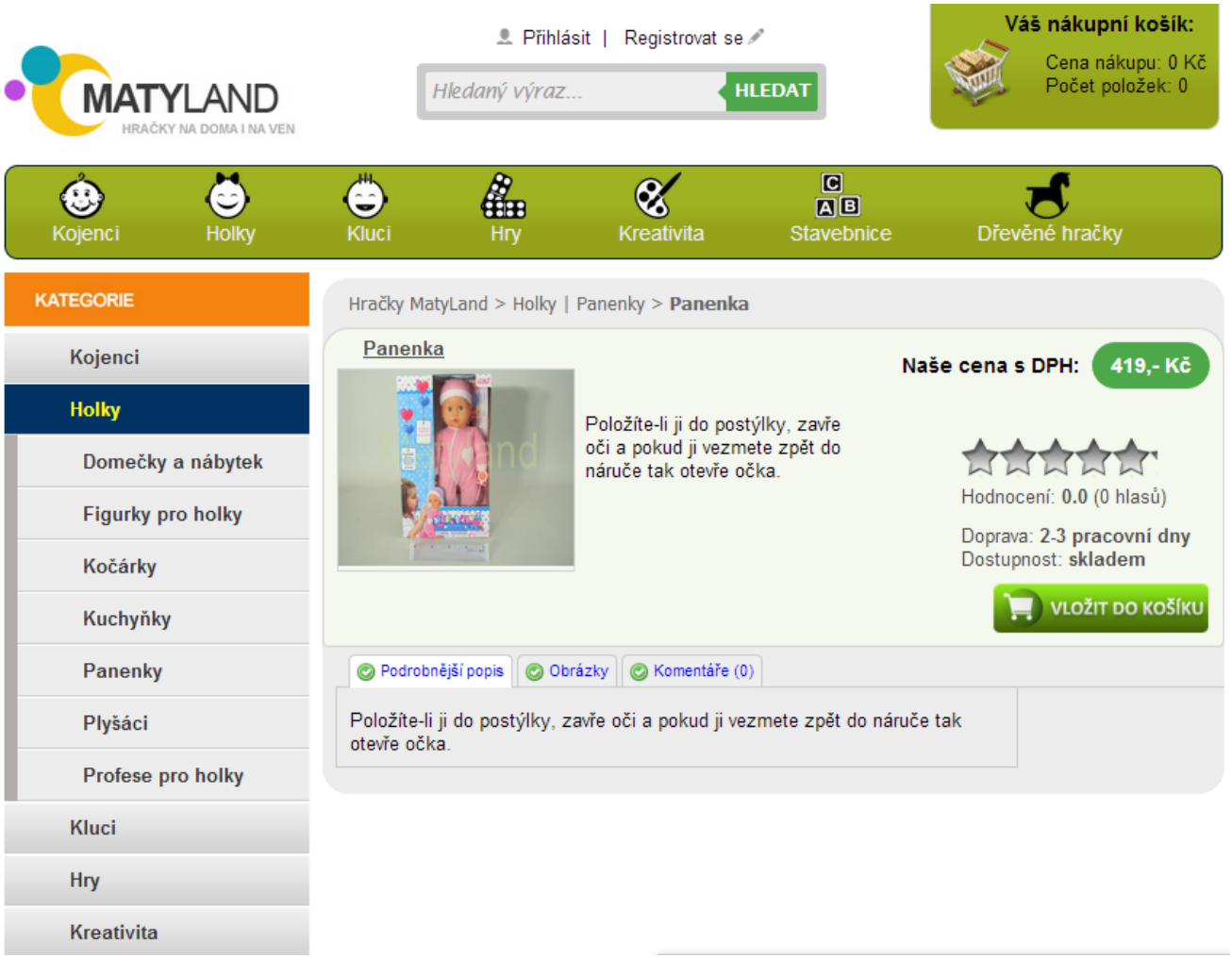
- jsem byl(a) seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo;
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, bakalářskou práci užít (§ 35 odst. 3);
- souhlasím s tím, že bakalářská práce bude v elektronické podobě archivována v Ústřední knihovně VŠB-TUO a jeden výtisk bude uložen u vedoucího bakalářské práce. Souhlasím s tím, že bibliografické údaje o bakalářské práci budou zveřejněny v informačním systému VŠB-TUO;
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona;
- bylo sjednáno, že užít své dílo, bakalářskou práci, nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Ostravě dne 10.5.2013

…….....
Jaroslav Klimčík

Seznam příloh

Příloha 1: Náhled stránky s produktem internetového obchodu



Příloha č. 2 - Náhled základních údajů v klientském centru

Příloha 3: Náhled detailu objednávky v administraci internetového obchodu